Aalto University
School of Engineering

Niko Braam

Automated Three-Dimensional Rotordynamic Simulation of Rotating Electrical Machines

Thesis submitted in partial fulfillment of the requirements for the degree of Master of Science in Technology.

Espoo, March 7th, 2016

Supervisor: Professor Kari Tammi, Aalto University
Instructors: Janne Roivainen, D.Sc. (Tech.)
Mikko Lyly, D.Sc. (Tech.)

Author Niko Braam

Title of thesis Automated Three-Dimensional Rotordynamic Simulation of Rotating Electrical Machines

Degree programme Mechanical Engineering

Major Machine Design **Code** K3001

Thesis supervisor Professor Kari Tammi

Thesis advisors Janne Roivainen, D.Sc. (Tech.), Mikko Lyly, D.Sc. (Tech.)

Date 07.03.2016 **Number of pages** 67+5 **Language** English

Abstract

Electric motors and generators are utilized widely in industrial applications. The design process of these rotating electrical machines often requires conducting well-known rotordynamic analyses. They include eigenvalue analyses to estimate the natural frequencies, mode shapes, and critical speeds of the machines to avoid harmful resonances. Conducting these analyses is often a routine process for analysts', and they can be automated. This frees analysts' time to solve more challenging tasks and reduces costs.

This thesis primarily aimed to create a model generator which generates three-dimensional models and finite element meshes of rotating electrical machine rotors based on product data. Furthermore, the aim was integration of the model generator with an analysis software to automate the whole model generation and eigenvalue analysis process.

To gain an understanding of the topic, this thesis reviews the advantages and difficulties of integrating geometry modeling, mesh generation and analysis as well as the basic theory of rotordynamic analysis. Furthermore, substructuring is presented as a method to increase flexibility and reduce computation time.

As a result of this thesis, a model generator was created and integrated with an analysis software. Two of the generated models were validated by comparing them to reference models. The natural frequencies were then calculated with these models and compared to experimentally measured values of the corresponding rotors. The first four free rotor natural frequencies were predicted with satisfactory accuracy, and the first critical speed was calculated with good accuracy. When compared to manual modeling and analysis work, the total time required to obtain free rotor natural frequencies and mode shapes was reduced to approximately one tenth. Further development of the created modeling and analysis system is recommended based on the results of this thesis.

Keywords rotordynamics, model generator, design automation, eigenvalues

Tekijä Niko Braam

Työn nimi Pyörivien sähkökoneiden automatisoitu kolmiulotteinen roottoridynaaminen simulointi

Koulutusohjelma Konetekniikka

Pääaine Koneensuunnittelu **Koodi** K3001

Työn valvoja Professori Kari Tammi

Työn ohjaajat TkT Janne Roivainen, TkT Mikko Lyly

Päivämäärä 07.03.2016 **Sivumäärä** 67+5 **Kieli** Englanti

Tiivistelmä

Sähkömoottoreita ja -generaattoreita käytetään useissa teollisissa sovelluskohteissa. Usein näiden pyörivien sähkökoneiden suunnitteluprosessin aikana suoritetaan hyvin tunnettuja roottoridynaamisia analyyseja. Näihin sisältyy ominaisarvoanalyysejä koneiden ominaistaajuuksien, ominaismuotojen ja kriittisten pyörimisnopeuksien arvioimiseen haitallisten resonanssien välttämiseksi. Kyseisten analyysien suorittaminen on usein rutiininomainen prosessi asiantuntijoille ja ne voidaankin automatisoida. Näin voidaan vapauttaa asiantuntijoiden aikaa vaativampien ongelmien ratkaisemiseen ja säästää kustannuksissa.

Tämän diplomityön ensisijainen tavoite oli luoda pyörivien sähkökoneiden roottoreiden kolmiulotteisia malleja ja elementtiverkkoja tuottava malligeneraattori, joka toimii tuotetiedon pohjalta. Tavoitteena oli lisäksi integroida malligeneraattori toimimaan laskentaohjelmiston kanssa koko mallinnus- ja ominaisarvoanalyysiprosessin automatisoimiseksi.

Työn teoriaosassa esitellään mallien luomisen, verkotuksen ja analyysien integroimisen hyötyjä ja esteitä sekä roottoridynaamisen analyysin taustateoriaa aihepiirin ymmärtämisen tueksi. Lisäksi alirakennetekniikka esitetään menetelmänä laskenta-ajan lyhentämiseen ja joustavuuden lisäämiseen.

Työn tuloksena luotiin automaattinen malligeneraattori, joka integroitiin toimimaan laskentaohjelmiston kanssa. Kaksi generoitua mallia validoitiin vertaamalla niitä referenssimalleihin. Näillä malleilla laskettuja ominaisarvoja verrattiin vastaavien roottoreiden kokeellisesti mitattuihin ominaisarvoihin. Ensimmäiset neljä vapaan roottorin ominaistaajuutta pystyttiin ennustamaan tyydyttävällä tarkkuudella ja ensimmäinen kriittinen pyörimisnopeus hyvällä tarkkuudella. Tarvittava kokonaisaika vapaan roottorin ominaistaajuuksien ja -muotojen selvittämiseen väheni noin kymmenesosaan kun verrataan manuaaliseen mallinnus- ja analyysityöhön. Työn tulosten perusteella voidaan suositella luodun mallinnus- ja laskentasysteemin jatkokehitystä.

Avainsanat roottoridynamiikka, malligeneraattori, suunnitteluautomaatio, ominaisarvot

Acknowledgements

This thesis was written between September 2015 and February 2016 at ABB Motors & Generators in Pitäjänmäki, Helsinki, Finland.

Firstly I would like to thank M.Sc. Jan Westerlund and M.Sc. Jari Jäppinen from ABB Motors & Generators for the opportunity to write this thesis about such an interesting and challenging topic.

Special thanks go to my thesis advisors D.Sc. Janne Roivainen and D.Sc. Mikko Lyly for guiding me through this project and providing support on the way. I would also like to thank my thesis supervisor Professor Kari Tammi for advice and support. For a nice working atmosphere I want to thank all colleagues working in the advanced calculations team of ABB.

Finally I wish to express my gratitude for the support of my family and friends during my studies.

Helsinki, Finland, March 7th, 2016

_____

Niko Braam

Contents

Symbols

$m, [M]$	Mass and mass matrix
$\ddot{x}, \dot{x}, x$	Acceleration, velocity and displacement in the x-direction
X, Y	Amplitudes of vibration in the x- and y-directions
$\ddot{y}, \dot{y}, y$	Acceleration, velocity and displacement in the y-direction
$\ddot{q}, \dot{q}, q$	$Nx1$ vector of accelerations, velocities and displacements
$k, k_x, k_y, [K]$	Stiffness, stiffness in x- and y- direction and stiffness matrix
i	Imaginary unit or direction (x, y, z), depending on the context
j	Direction (x, y, z)
e	Napier's constant
ω	Frequency
t	Time
c	Viscous damping coefficient
d	Hysteretic damping coefficient
q_0	$Nx1$ vector of time-independent amplitudes
ω_r	Natural frequency of the r^{th} mode
$\{\Psi\}_r$	Mode shape of the r^{th} mode
$i[D]$	Hysteretic damping matrix
η_r	Damping loss factor for the r^{th} mode
λ_r	r^{th} eigenvalue
$\dot{\varphi}$	Whirl speed
β	Angle
Ω, Ω_z	Rotational speed, rotational speed around the z-axis
$[G(\Omega)]$	The gyroscopic matrix
L	Length
J	Polar moment of inertia
I_0	Moment of inertia
$\dot{\theta}$	Angular velocity
M_x	Moment about the x-axis
ε	Strain
$\varepsilon_{xx}, \varepsilon_{yy}, \varepsilon_{zz}$	Strains in cartesian coordinates
$2\varepsilon_{xy}, 2\varepsilon_{yz}, 2\varepsilon_{zx}$	Shear strains in cartesian coordinates
σ	Stress
$\sigma_{xx}, \sigma_{yy}, \sigma_{zz}$	Stresses in cartesian coordinates
$\sigma_{xy}, \sigma_{yz}, \sigma_{zx}$	Shear stresses in cartesian coordinates
δ	Elasticity tensor
G	Shear modulus
G_{xy}, G_{yz}, G_{zx}	Shear moduli in cartesian coordinates
E	Young's modulus
E_{xx}, E_{yy}, E_{zz}	Young's moduli in cartesian coordinates
v	Poisson's ratio
v_{xy}, v_{yz}, v_{zx}	Poisson's ratios in cartesian coordinates

Abbreviations

1D	One-dimensional
2D	Two-dimensional
3D	Three-dimensional
ABB	Asea Brown Boveri – A Multinational Engineering Corporation
BREP	Boundary representation
CAD	Computer-aided design
CAE	Computer-aided engineering
CFD	Computational fluid dynamics
CMS	Component mode synthesis
D-END	Drive end of a shaft
DOF	Degree of freedom
DOL	Diameter in the left end of a segment
DOR	Diameter in the right end of a segment
ELEN	Element length
FEA	Finite element analysis
FEM	Finite element method
GEOM	Geometry module of Salome
GUI	Graphical user interface
ID	Identifier
MDOF	Multi-degree-of-freedom
NEL	Element identification number
N-END	Non-drive end of the shaft
NX	CAD-program developed by Siemens PLM Software
PBS	Portable batch system
RPM	Revolutions per minute
SDOF	Single-degree-of-freedom
SMESH	Meshing module of Salome
TUI	Text user interface
UNIX	Trademark of computer operating systems

1 Introduction

1.1 Background

Rotating electrical machines, such as motors and generators are utilized widely in many industrial applications. The design process of them often requires conducting many well-known structural dynamic analyses. These include eigenvalue analyses to estimate the natural frequencies and critical speeds of the machines to avoid harmful resonances. Conducting these analyses is often a routine process for analysts'. The whole process can be automated and parametrized with reasonable effort by utilizing the computational processing power and software capabilities available.

In this context, automation means that minimum human activity is required to perform the geometry modeling and analyses. This frees analyst's time to solve more complicated tasks which reduces design lead time and costs. Moreover, the availability of reliable and customizable open-source engineering tools allows minimizing the licensing costs. Combined with in-house programming these tools enable flexible modifications when compared to commercial softwares and closed-source code. Furthermore, close integration of geometry, meshing, and analysis enables design optimization and reduces errors caused by humans.

This thesis is written as part of a project at ABB (Asea Brown Boveri) Oy. In the long term the project aims to automate routine structural dynamic analyses. An additional goal is the replacement of a commercial two-dimensional (2D) rotordynamic analysis software *Ardas* with a more accurate three-dimensional (3D) analysis system. A further goal is to include all main components, such as bearings, stators and rotors, of rotating electrical machines in dynamic analyses through coupling them together with substructuring. The benefit of substructuring is that the acquired dynamic models typically have less than 100 000 degrees of freedom (DOFs) while full models without substructuring typically have over 10 million DOFs. Applying substructuring speeds up the computation process by reducing the amount of DOFs while simultaneously increasing flexibility. If one component of a large system is modified, only that particular component has to be fully analyzed again rather than the full system. The full system dynamics can then be assessed by including the dynamics of the modified component.

1.2 Research problem and objective

This study addresses how to create 3D finite element meshes and run eigenvalue analyses automatically for rotors of rotating electrical machines. Creating a prototype of a software module which fulfills the task of geometry and mesh generation by utilizing product data is the primary objective of this study. Full automation of the mesh generation and analysis

process requires a continuous flow of information inside the process. This emphasizes the importance of consistent and reliable topology management of the generated meshes especially when combining different engineering tools. The information to be passed inside the process includes data to assign the material parameters and boundary conditions. In addition, topology management is important for post-processing the results and utilization of substructuring.

The validation of two generated geometries by comparing them to reference geometries is the secondary objective of this study. These geometries are then utilized to calculate free rotor natural frequencies which are then compared to experimentally measured values of the corresponding real machines. In addition, they are compared to results calculated with *Ardas*. Moreover, a Campbell diagram for one rotor model is calculated and compared to a Campbell diagram calculated with *Ardas*. The purpose of the comparisons is to gain an understanding of the accuracy of the created analysis system prototype without any further adjustments to increase the accuracy.

1.3 Scope of the study

This study focuses on automatically creating 3D finite element meshes of rotating electrical machine rotors for use in eigenvalue analysis. Developing the numerical methods applied for calculating the results is out of the scope of this study. Furthermore, adjusting the material and computation parameters to minimize the error between calculated and measured values is also out of the scope of this study. Moreover, all experimental measurement values and values calculated with *Ardas* are acquired from ABB because no experimental measurements or simulations with other software are conducted as part of this thesis.

1.4 Research methods

This thesis is divided in three main sections. Section 2 consists of the theory background review and the creation of the model generator. Section 3 discusses the results of geometry validation and analysis value comparisons. Section 4 summarizes the study and presents suggestions for future work.

Firstly, the advantages and difficulties of integrating geometry modeling, mesh generation and analysis are reviewed in chapter 2.1. To gain an understanding of the basic concepts of rotordynamic analysis, the background theory related to eigenvalue analysis and estimating rotor critical speeds is reviewed in chapter 2.2. Stiffness matrix calculation is presented in subchapter 2.2.4, because it is required to define the correct material data input for the analysis software. Furthermore, substructuring with component mode synthesis is shortly introduced as a method to speed up analyses and increase flexibility in subchapter 2.2.5.

The requirements of the model generator are defined in chapter 2.3. Then the creation process of the model generator is discussed in chapter 2.4. After creating the model generator, two generated geometries are compared to reference geometries in chapter 3.1 and the calculated eigenvalue analysis results are compared to reference results in chapters 3.2 and 3.3.

The practical section of this thesis covers many aspects. They are accomplished with the help of in-house knowledge, open-source engineering tools and user documentation. The platform for geometry creation and meshing is *Salome*, which is a customizable open-source software that provides a generic platform for generation of geometry and preparing it for numerical calculations. An open-source tetrahedral mesh generator *NETGEN* is utilized for meshing since it functions inside *Salome*. The programming is conducted with Python because it enables commanding *Salome* directly through the text user interface (TUI). Moreover, an open-source multi-physics simulation software *Elmer* is applied for numerical calculations and *Octave*, a language for numerical calculations is utilized for stiffness matrix calculation.

2 Automated computational engineering

This section discusses the materials and methods applied in this thesis. The first chapter 2.1 discusses advantages and difficulties in integration of computer-aided design (CAD) and computer-aided engineering (CAE). The second chapter 2.2 introduces the theoretical background of rotordynamic analysis. Chapters 2.3 and 2.4 discuss the definition and creation of the model generator.

2.1 Integration of CAD and CAE

Close integration of CAD and CAE is crucial to improve the product design process. Finite element analysis (FEA) is one of the most common CAE methods, but often models created with CAD are not suitable for FEA without additional modifications. FEA programs usually require simplified models while CAD programs often provide detailed models. Therefore, to use CAD models in FEA, an appropriate idealization process is often required to remove unnecessary details from the CAD model. This process is usually non-intuitive and time-consuming, which is a notable obstacle to the integration of CAD and CAE. (Lee 2005, p. 941)

It has been studied at Sandia National Laboratories, that of the overall analysis process time approximately 60% is spent for analysis geometry creation, 20% for mesh generation, and only 20% of the total time is dedicated for analysis as shown in Figure 1. The 80/20 modeling/analysis ratio appears to be very common in the industry. The shortcomings of close integration of geometry modeling, meshing and analysis in current engineering procedures inhibit the utilization of cutting edge technologies such as design optimization. To enable for example shape optimization, the CAD geometry-to-mesh mapping has to be automatic, differentiable and closely integrated with the analysis and optimization software. (Cottrell *et al*. 2009, p. 1-3)

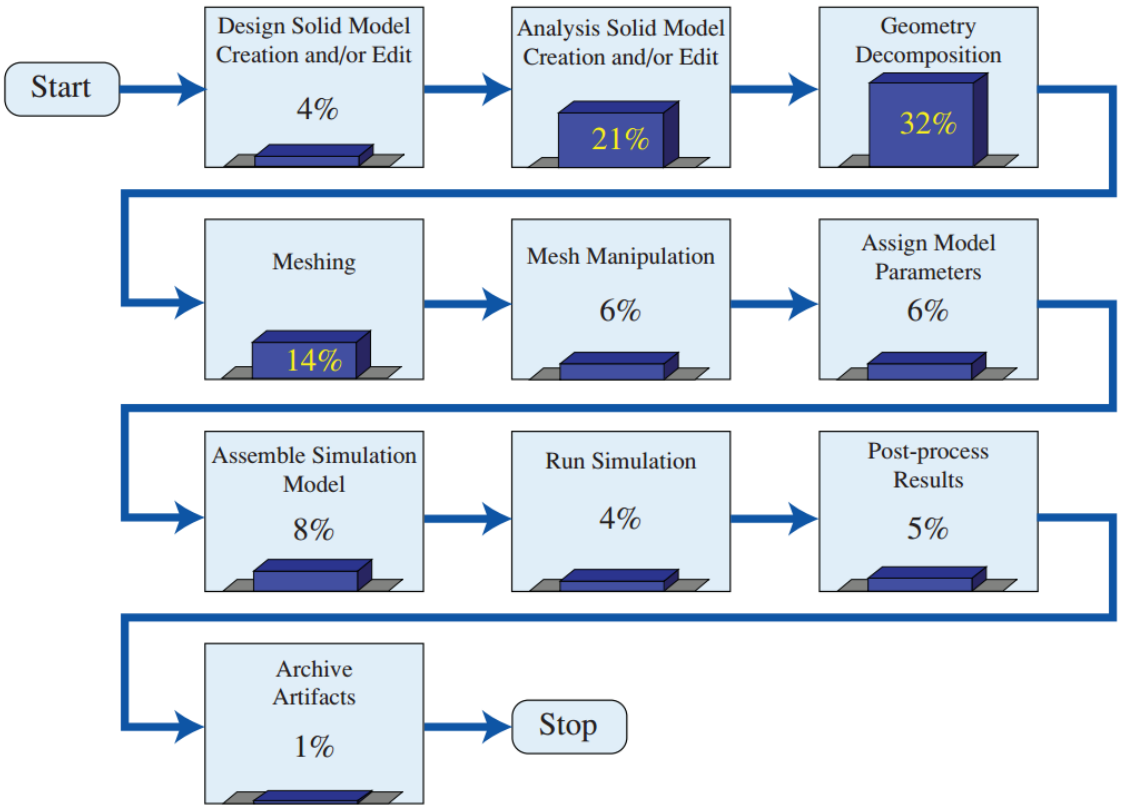

Figure 1. Estimation of the relative time required for each task in the model creation and analysis process at Sandia National Laboratories. The time spent building the analysis model has the greatest share of the total time. (Cottrell et al. 2009, p. 3)

Many research efforts have been done to automate the CAD geometry simplification process. Still, only limited automated processes which require further improvement exist. However, in the CAE-centric design process, engineering analyses are performed initially using idealized analysis models instead of first creating a detailed CAD model and then simplifying it for analysis. The design concept is then defined and refined based on the analyses before establishing the full CAD product model as shown in Figure 2. (Lee 2005, p. 941-942) In conclusion, the CAE-centric design approach allows avoiding the CAD geometry simplification process in most cases.

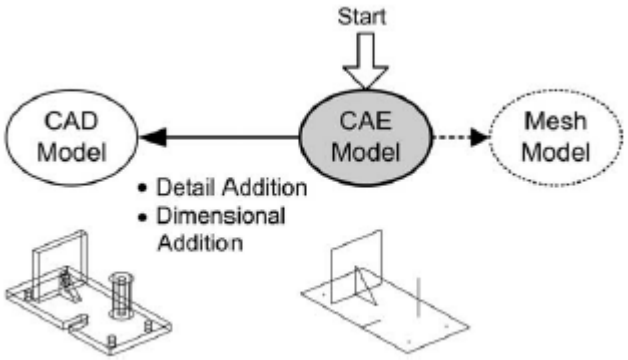

Figure 2. The CAE-centric design approach. The detailed CAD model is created after creating and analyzing the simplified CAE model. (Lee 2005, p. 942)

For example, Rodriguez & Sturdza (2006) have developed a new platform-independent java-based rapid geometry engine which applies parametric geometry generation for aerodynamic analysis in preliminary aircraft design. The geometry engine allows the designer to bypass laborious geometry modeling and directly analyze the geometry. (Rodriguez & Sturdza, 2006) Also Takenaka *et al.* (2000) have developed a system to automatically create geometries of exhaust ports using the programming functionality of a commercial CAD system. A computational fluid dynamics (CFD) code is then applied to automatically generate a mesh based on the CAD geometry and to predict the port performance. Utilization of this technique reduced the time to develop a port to one tenth of the time required by the conventional method without taking the computation time into account. (Takenaka *et al.*, 2000)

There are a number of geometry problems which can prevent the benefits of the most advanced automatic meshing algorithms or at least impair their ability to generate good quality meshes. (Jones *et al.* 1995, p. 1) Two of the most common causes along with ways to avoid these problems are discussed in the next paragraphs.

Gaps are locations where the topology of a solid or surface is incomplete. For example, edges or faces can be unconnected. There are two possible solutions to this problem. If the gap is small, the adjacent vertices, edges, or faces can be merged. When the gap is big and the previous solution is not optimal, one solution is to grow the adjacent faces to cover the gap. Multiple definitions occur when there are multiple definitions of the same vertex, edge, or face. These may appear topologically unconnected and cause gaps. In this case, the solution is also merging the duplicate definitions to a single definition. (Jones *et al.* 1995, p. 3-4)

Tangencies and slivers shown in Figure 3 occur when there are very small faces or sharp angles on faces. These might be caused by edges meeting at very small angles or long and narrow surfaces. Sliver surfaces can be removed by merging the points and lines together

producing a single edge. The same can be applied to tangencies by merging a portion of the tangential edges. (Jones *et al.* 1995, p. 4-5)

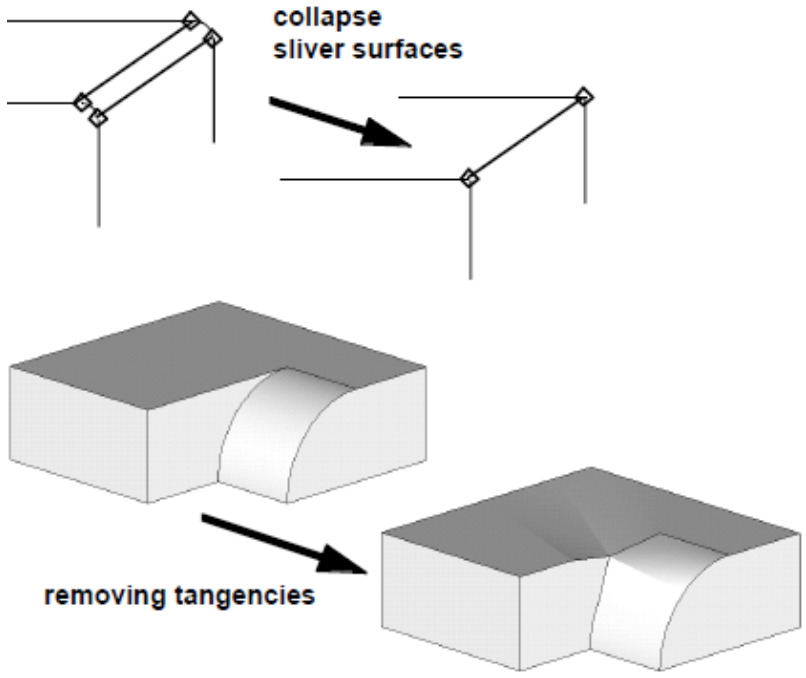

Figure 3. Sliver surfaces and tangencies are common geometry problems which prevent the benefits of automatic meshing algorithms or at least impair their ability to generate good quality meshes. (Modified from Jones et al. 1995, p. 4-5)

2.2 Rotordynamic analysis

This thesis focuses on two of the main objectives of rotordynamic analysis which are eigenvalue analysis and estimating critical speeds. The theoretical background related to those objectives is presented in subchapters 2.2.1, 2.2.2 and 2.2.3. Stiffness matrix calculation is discussed in subchapter 2.2.4, because it is required to define the solver input file of *Elmer*. Additionally, one method of substructuring rotor-bearing systems using component mode synthesis is shortly presented in subchapter 2.2.5. Topics related to rotor instability and response analysis are out of scope of this thesis. Figure 4 shows an exploded view of an electric motor with the main components relevant to this thesis.

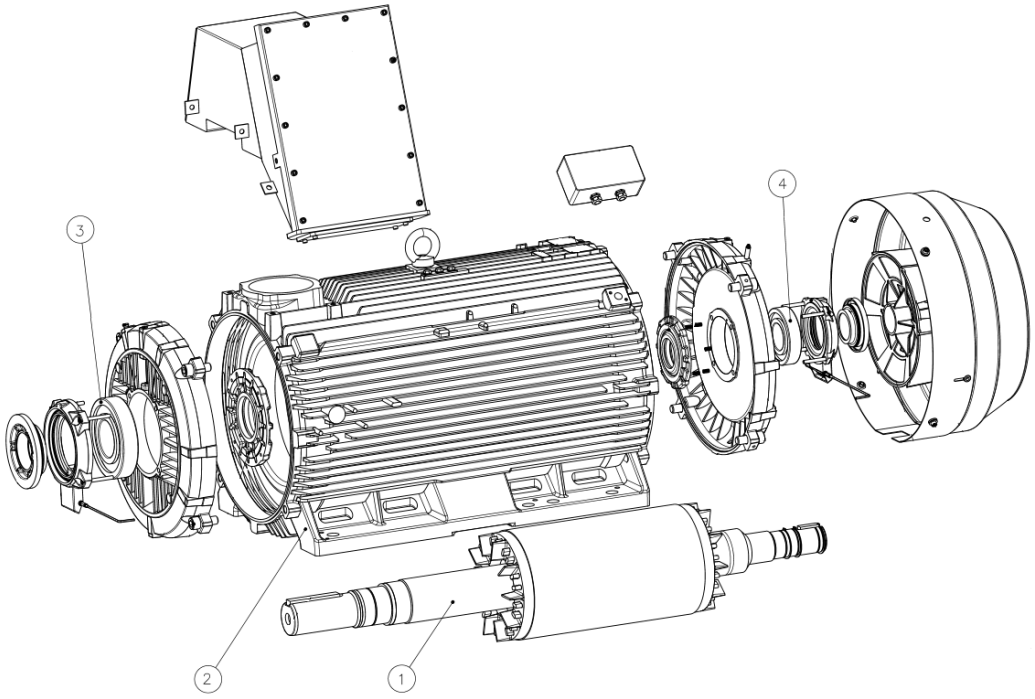

Figure 4. Exploded view of an electric motor. The numbered parts are: 1. Rotor, 2. Stator frame (stator inside), 3. Drive end (D-end) bearing, 4. Non-drive end (N-end) bearing. (Modified from ABB 2011, p. 46)

2.2.1 Non-rotating single-degree-of-freedom systems

Properties of a linear single-degree-of-freedom (SDOF) system are important, because the properties of a linear multi-degree-of-freedom (MDOF) system can be represented as the superposition of a number of SDOF characteristics. Only a few practical systems can be modeled by a SDOF system. There are three generally known classes of a system model (Ewins 2000, p. 28):

- undamped
- viscously-damped
- hysteretically- (or structurally-) damped

Figure 5 shows the basic model for the SDOF system where $f(t)$ and $x(t)$ are general time-varying force and displacement quantities. The spatial model consists of a mass (m) and a spring (k) and when damped either a viscous dashpot (c) or a hysteretic damper (d) is included. (Ewins 2000, p. 28-29)

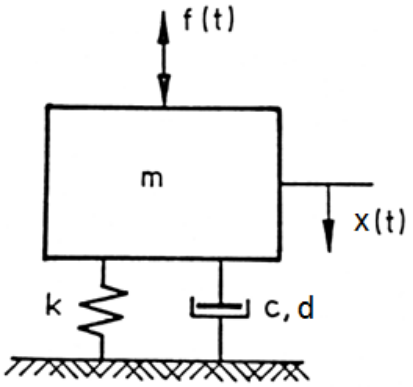

Figure 5. Single-degree-of-freedom (SDOF) system. (Modified from Ewins 2000, p. 28)

In case of an undamped system without external forces the governing equation of motion is (Ewins 2000, p. 29):

$$m\ddot{x} + kx = 0 \tag{1}$$

Assuming a harmonic motion and using the following trial solution (Ewins 2000, p. 29):

$$x(t) = Xe^{i\omega t} \tag{2}$$

where $i^2 = -1$, ω is the frequency, X is the amplitude of vibration and t is the time leads to the requirement that (Ewins 2000, p. 29):

$$k - \omega^2 m = 0 \tag{3}$$

Thus the model consists of a single mode of vibration with a natural frequency $\omega_0 = \sqrt{\frac{k}{m}}$. A natural frequency ω is the frequency at which a system tends to freely vibrate without any driving force. (Inman 2007, p. 8). By adding a viscous dashpot c, thus taking viscous damping into account, the equation of motion for free vibration becomes (Ewins 2000, p. 30):

$$m\ddot{x} + c\dot{x} + kx = 0 \tag{4}$$

In this case a more general trial solution has to be applied (Ewins 2000, p. 30):

$$x(t) = Xe^{st} \tag{5}$$

with which the condition that must be satisfied for a solution to exist can be obtained:

$$(ms^2 + cs + k) = 0 \tag{6}$$

which leads to

$$s_{1,2} = -\omega_0 \xi \pm \mathrm{i}\omega_0 \sqrt{1 - \xi^2} \tag{7}$$

where ${\omega_0}^2 = \frac{k}{m}$ and $\xi = \frac{c}{2\sqrt{km}}$.

By substituting equation (7) to (5), a following modal solution is obtained (Ewins 2000, p. 30):

$$x(t) = Xe^{-\omega_0 \xi t} e^{i\left(\omega_0 \sqrt{1-\xi^2}\right)t} \tag{8}$$

which represents a single mode of free vibration with a complex natural frequency having two parts: an imaginary or oscillatory part with a frequency $\omega_0 \sqrt{1 - \xi^2}$ and a real or decay part with a damping rate $\xi \omega_0$.

Inspection of real structures shows that the viscous damping model utilized above is not very representative when applied to MDOF systems. There is a frequency-dependence exhibited by real structures which is not described by the viscous damping model. All structures have a degree of damping due to the hysteresis properties of the materials from which they are made. An alternative theoretical damping model is provided by the hysteretic damper, also known as the structural damper. It satisfies the requirement of an effective damper rate which varies inversely with frequency, thus $c = \frac{d}{\omega}$ where d is the hysteretic damping coefficient. It also satisfies the requirement that the energy lost per cycle is independent of frequency. (Ewins 2000, p. 32-33) Hysteretic damping can be modeled by modifying the traditional linear stress-strain curve into an elliptical hysteresis cycle as shown in Figure 6. (Genta 2004, p. 57)

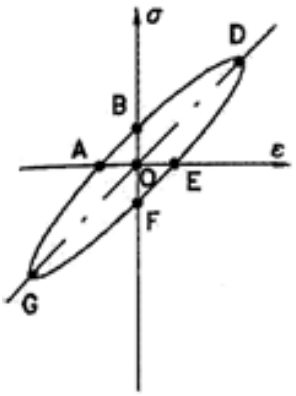

Figure 6. The hysteresis cycle of a material in the σε-plane. (Modified from Genta 2004, p. 57)

The area between the curve ABD and the ε-axis is the energy supplied to the material during the loading phase. The area under the curve DE is the energy given back by the material during unloading. Thus, the total area of the ellipse is the energy dissipated by the hysteresis during one cycle. The ellipse is so narrow for structural materials that it can hardly be distinguished from the straight line OD. (Genta 2004, p. 57) Another common source of energy dissipation in practical structures is the friction which exists in joints between the components of the structure. (Ewins 2000, p. 32)

2.2.2 Non-rotating multi-degree-of-freedom systems

For an undamped MDOF system without external forces with N degrees of freedom, the governing equations of motion can be written in matrix form as follows (Ewins 2000, p. 50):

$$[M]\{\ddot{q}(t)\} + [K]\{q(t)\} = \{0\} \tag{9}$$

where $[M]$ and $[K]$ are NxN mass and stiffness matrices, respectively, and $\{q(t)\}$ is a $Nx1$ vector of time-dependent displacements.

Again, by assuming a trial solution of the form (Ewins 2000, p. 51):

$$\{q(t)\} = \{q_0\}e^{i\omega t} \tag{10}$$

where $\{q_0\}$ is a $Nx1$ vector of time-independent amplitudes for which it is clear that $\{\ddot{q}\} = -\omega^2\{q_0\}e^{i\omega t}$ and $i^2 = -1$. Substituting the trial solution (10) to the equation of motion (9) leads to (Ewins 2000, p. 51):

$$([K] - \omega^2[M])\{q_0\}e^{i\omega t} = \{0\} \tag{11}$$

for which the only non-trivial solutions are those for which:

$$det([K] - \omega^2[M]) = 0 \tag{12}$$

N values of the systems undamped natural frequencies ω^2 can be found by solving equation (12). Substituting any of these back to equation (11) yields the corresponding set of values for $\{q_0\}$ also noted as $\{\Psi\}_r$, the so-called mode-shapes of the corresponding natural frequencies. When the system vibrates with its natural frequency, it tends to deform to the corrensponding mode shape (Inman 2007, p. 270-279). The complete solutions can be written with two NxN matrices as (Ewins 2000, p. 51):

$$[\omega_r^2]\,,[\Psi_r] \tag{13}$$

where ω_r^2 is the rth eigenvalue or natural frequency squared and Ψ_r, the eigenvector matrix, is a description of the corresponding eigenmode or mode shape. These two eigenmatrices constitute the modal model and $[K]$ and $[M]$ constitute the spatial model. It is important to note that the eigenvalue matrix is unique, but the eigenvector matrix is not. It can be scaled with any factor. This does not affect the mode shape itself, but only its amplitude. (Ewins 2000, p. 51-52)

Applying hysteretic damping to an undamped system without external forces leads to the following equation of motion (Ewins 2000, p. 66):

$$[M]\{\ddot{q}(t)\} + ([K] + i[D])\{q(t)\} = \{0\} \tag{14}$$

where $i[D]$ is the hysteretic damping matrix. A following trial solution can be assumed (Ewins 2000, p. 66):

$$\{q(t)\} = \{q_0\}e^{i\lambda t} \tag{15}$$

where λ is allowed to be complex. Substituting this trial solution (15) to the equation of motion (14) leads to a complex eigenproblem whose solution is also in the form of two eigenmatrices as in the preceding example without damping and external forces. In this case, however both eigenmatrices are complex. The rth eigenvalue can be written as (Ewins 2000, p. 66-67):

$$\lambda_r^2 = \omega_r^2(1 + i\eta_r) \tag{16}$$

where ω_r is the is the natural frequency and η_r is the damping loss factor for the respective mode. The natural frequency is not necessarily equal to the corresponding natural frequency of the undamped system, although the two values are generally very close to each other. (Ewins 2000, p. 67)

In the case of complex modes shown in Figure 7, each DOF can be considered as having both a magnitude and a phase angle. This is only very slightly different from the undamped case, where each DOF has a magnitude and a phase angle which is either 0˚ or 180˚, both of which can be entirely described with real numbers. The inclusion of general damping generalizes this feature to allow the phase angle of each DOF to be anything between 0˚ and 180˚. This means that each DOF of a structure reaches its own maximum deflection at a different instant in the vibration cycle as its neighbouring DOFs which all have different phases. On the contrary, a real mode is one in which all phase angles are equal and each DOF of a structure reaches their own maxima at the same instant in time. This holds true also for the zero deflection positions, which means that there are two time instants when the structure is completely undeformed. This is not a property of a complex mode, since the zero deflection positions of a complex mode are also reached at different time instants. As a conclusion, a real mode has the appearance of a standing wave while a complex mode can better be described as exhibiting travelling waves. (Ewins 2000, p. 67, p. 113)

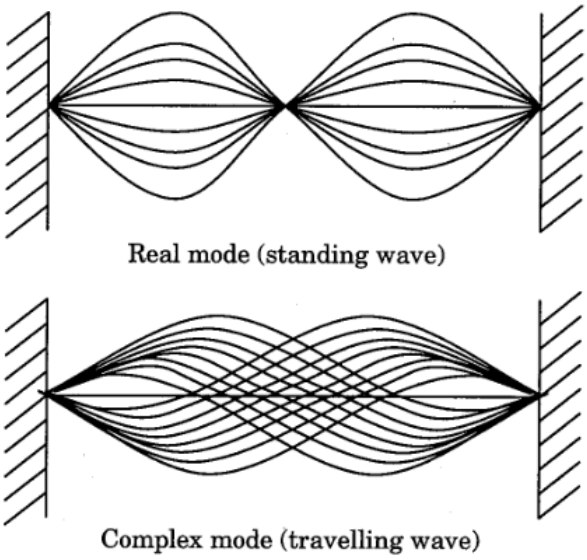

Figure 7. A real and complex mode of a beam fixed from both ends. (Ewins 2000, p. 114)

2.2.3 Rotating multi-degree-of-freedom rotor systems

The great majority of rotordynamic problems related to rotating MDOF rotor systems involve synchronous whirling, *i. e.* response to imbalance, since real machines never have a perfectly balanced rotor. Also, nonsynchronous whirling exists, which is the more destructive whirling related to instability. However, instability problems are out of scope, and therefore those are not discussed further. Figure 8 shows an end-view of a forward whirling rotor. The black element represents an unbalanced mass. $\dot{\varphi}$ is the whirl speed which is equal to the rotational speed Ω_z. The rotor imbalance vector U leads the whirl

vector V by a constant angle β which is the characteristic of synchronous whirling. Whirling causes the geometric center point of the shaft to move in an orbit around the inertial coordinate system origin. (Vance 1987, p. 4-7)

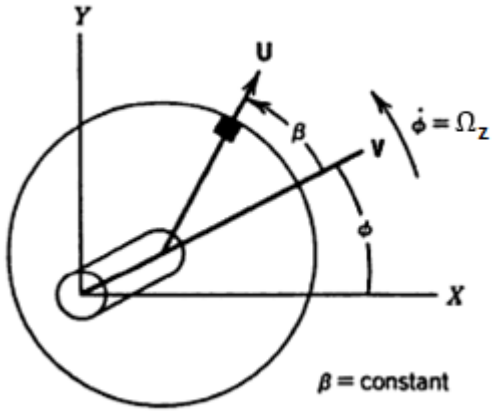

Figure 8. Synchronous whirling. The rotor imbalance vector U leads the whirl vector V by a constant angle β. (Modified from Vance 1987, p. 7)

A simple linear 2 DOF system shown in Figure 9 is presented to approach the equations of more general types of rotating structures. All equations are written in the stationary reference frame xyz. They can also be written in the rotating reference frame. (Ewins 2000, p. 82-83)

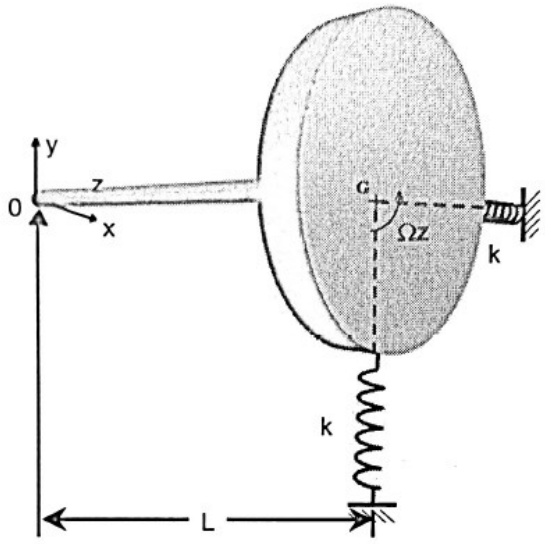

Figure 9. A simple 2 DOF rotor system. (Ewins 2000, p. 84)

The system consists of a rigid disc mounted on the free end of a rigid shaft of length L. The other end is pin-jointed by a rigid bearing. At the free end the shaft is supported by a

flexible bearing, described by vertical and horizontal stiffnesses, k_x and k_y, respectively. The polar moment of inertia of the disc and shaft is J. The moment of inertia of the disc about a lateral axis through the origin 0 is I_0. The rotor has a rotational speed of Ω_z and no damping or external excitation. When the disc is vibrating in the x and/or y directions, it is simultaneously rotating about more than one axis. This motion causes Coriolis accelerations which are usually referred to as gyroscopic forces. Rotation about the z-axis with a rotational speed Ω_z and about the y-axis with an angular velocity $\dot{\theta} = \dot{x}/L$ can only exist if there is a moment with a magnitude $M_x = J\Omega_z\dot{x}/L$ applied to the system about the x-axis. The moment M_x and its counterpart for another combination of rotation has the effect of coupling the two equations of motion, which now take the following form (Ewins 2000, p. 83-85):

$$\begin{bmatrix} I_0/L^2 & 0 \\ 0 & I_0/L^2 \end{bmatrix}\begin{Bmatrix} \ddot{x} \\ \ddot{y} \end{Bmatrix} + \begin{bmatrix} 0 & J\Omega_z/L^2 \\ -J\Omega_z/L^2 & 0 \end{bmatrix}\begin{Bmatrix} \dot{x} \\ \dot{y} \end{Bmatrix} + \begin{bmatrix} k_x & 0 \\ 0 & k_y \end{bmatrix}\begin{Bmatrix} x \\ y \end{Bmatrix} = \begin{Bmatrix} 0 \\ 0 \end{Bmatrix} \tag{17}$$

Again assuming harmonic motions in both directions as $x(t) = Xe^{i\omega t}$ and $y(t) = Ye^{i\omega t}$ and defining $k = k_x = k_y$ leads to the following linear equations (Ewins 2000, p. 86):

$$\begin{bmatrix} k - \omega^2 I_0/L^2 & i\omega J\Omega_z/L^2 \\ -i\omega J\Omega_z/L^2 & k - \omega^2 I_0/L^2 \end{bmatrix}\begin{Bmatrix} X \\ Y \end{Bmatrix} = \begin{Bmatrix} 0 \\ 0 \end{Bmatrix} \tag{18}$$

calculating the determinant leads to a characteristic equation (Ewins 2000, p. 86):

$$(kL^2)^2 - \omega^2\left(2I_0kL^2 + J^2{\Omega_z}^2\right) + \omega^4{I_0}^2 = 0 \tag{19}$$

which can be solved to find the natural frequencies ω_1 and ω_2 (Ewins 2000, p. 87):

$$\omega_1 = \omega_\Omega - \frac{1}{2}(\gamma\Omega_z) \tag{20}$$

$$\omega_2 = \omega_\Omega + \frac{1}{2}(\gamma\Omega_z) \tag{21}$$

where

$${\omega_0}^2 = \frac{kL^2}{I_0};\ \gamma = \frac{J}{I_0} \text{ and } {\omega_\Omega}^2 = {\omega_0}^2 + \frac{1}{4}(\gamma\Omega_z)^2$$

Completing the free vibrations solution shows that the mode shapes corresponding to the two natural frequencies are complex (Ewins 2000, p. 88):

$$\{\Psi\}_1 = \begin{Bmatrix} 1 \\ i \end{Bmatrix} \tag{22}$$

$$\{\Psi\}_2 = \begin{Bmatrix} i \\ 1 \end{Bmatrix} \tag{23}$$

The mode shape corresponding the first natural frequency ω_1 represents a motion which constitutes a circular orbit of the disc centre which is backwards with respect to Ω_z. The mode shape corresponding the second natural frequency ω_2 represents a forward circular orbiting motion in the same directions as Ω_z. The mode shapes are illustrated in Figure 10. (Ewins 2000, p. 87-88)

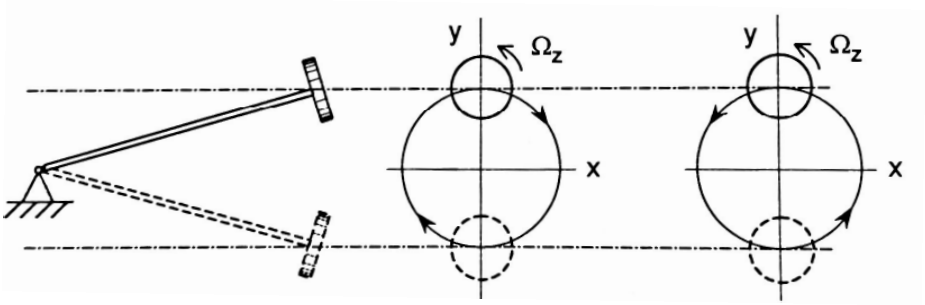

Figure 10. Mode shapes of the 2 DOF rotor system. The figure in the middle represents backward whirling and the rightmost figure represents forward whirling. (Modified from Ewins 2000, p. 88)

The gyroscopic moments caused by rotation tend to stiffen the rotor in forward whirling where $\dot{\varphi} > 0$ and to destiffen the rotor in backward whirling where $\dot{\varphi} < 0$. With increasing rotational speed this causes the natural frequencies to decrease in backward whirling and to increase in forward whirling. (Vance 1987, p. 122-123) Because the natural frequencies ω_r of the rotor system depend on the rotational speed Ω_z, they can be plotted as functions of Ω_z. Since the exciting forces also depend on the rotational speed, they can be illustrated on the same plot and the result is generally known as a Campbell diagram shown in Figure 11. In the case of synchronous unbalance, the excitation can be presented in the Campbell diagram by a straight line $\omega = \Omega_z$ through the origin which represents simple proportionality to the rotational speed. (Genta 2004, p. 9-14)

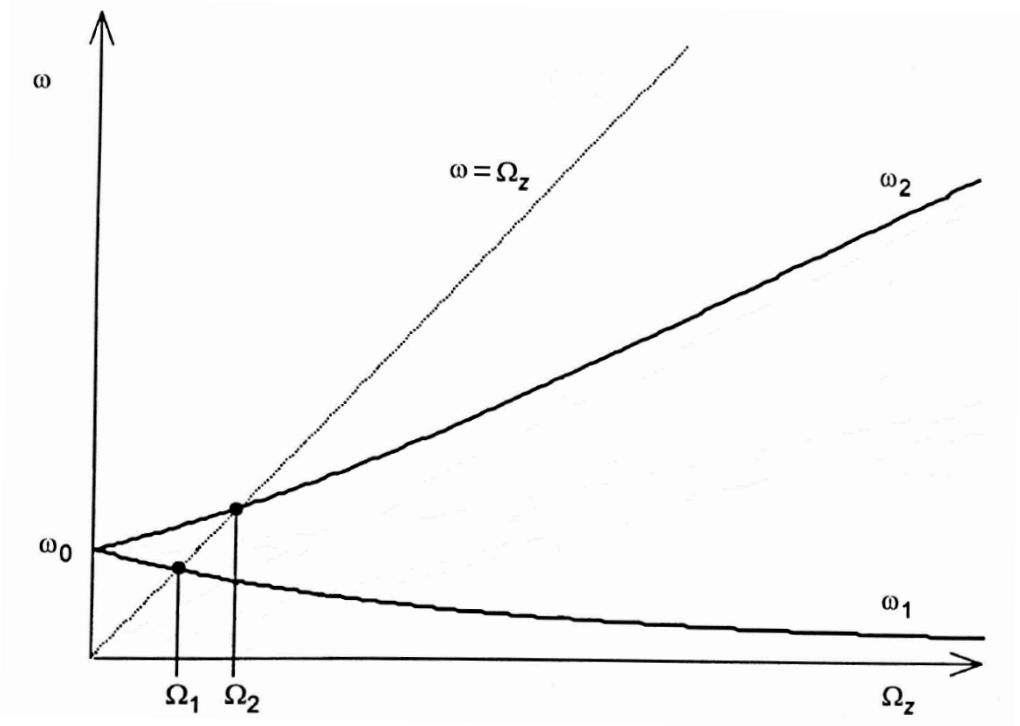

Figure 11. The Campbell diagram of a 2 DOF rotor system. ω_1 and ω_2 are the natural frequencies of backward and forward whirling, respectively. Ω_z is the rotational speed of the rotor. Ω_1 and Ω_2 denote the first two critical speeds. (Modified from Ewins 2000, p. 88)

A critical speed is called a rotational speed at which the rotor imbalance excites a natural frequency of the rotor system. When the rotor is revolving at a critical speed, the shaft is bowed in the mode shape corresponding with the natural frequency. (Vance 1987, p. 116) The critical speeds can be located from a Campbell diagram by observing the intersections of the curves describing natural frequencies and excitations. Thus in Figure 11 the critical speeds can be obtained by finding the intersections of the curves $\omega = \Omega_z$ and ω_r as functions of the rotational speed Ω_z. (Genta 2004, p. 15)

Not all critical speeds are equally dangerous. No actual resonance occurs if the excitation caused by the torsional driving moment of the rotor coincides with a natural frequency related to a flexural mode when flexural and torsional behavior are uncoupled. However, the situation where the excitation caused by unbalance coincides with one of the flexural natural frequencies is particularly dangerous. These critical speeds are usually referred as flexural critical speeds and the less dangerous ones are referred as secondary critical speeds. The rotor can not operate for long periods of time at or near flexural critical speeds without strong vibrations or even failure. However, there are rotors which are designed to operate between the two first critical speeds. In this case the first critical speed is passed. The ranges below and above the first critical speed are called the subcritical and supercritical range, respectively. (Genta 2004, p. 15-18)

When extending this simple 2 DOF system to more complex systems, the equations of motion for a more general type of a rotating system with hysteretic damping can be expressed in matrix form as follows (Ewins 2000, p. 102):

$$[M]\{\ddot{q}(t)\} + [G(\Omega)]\{\dot{q}(t)\} + ([K] + i[D])\{q(t)\} = \{0\} \quad (24)$$

In this equation the rotational speed dependent gyroscopic matrix $[G(\Omega)]$ is skew-symmetric, while all other matrices are symmetric. In addition, the stiffness matrix may depend on rotational speed (Genta 2004, p. 6). The solution of the equations follows different routes depending upon the specific features of each case. The eigensolution yields a single eigenvalue matrix, as usual, and two sets of complex eigenvectors. In this general case each complex eigenvalue consists of the frequency and damping for one mode of vibration and the two corresponding eigenvectors describe the mode shapes. (Ewins 2000, p. 102-103)

2.2.4 Stiffness matrix calculation

The solver input file of *Elmer* software requires the stiffness matrices of each material present in the rotor model to be defined separately. Because the rotor core is manufactured by stacking thin insulated electric sheets, its material properties cannot be modeled as isotropic. If a continuous material model is assumed, the material properties of the rotor core can be modeled as transversely isotropic (Kanninen 2006, p. 4-9). All other materials present in the rotor model are assumed to be isotropic. This subchapter introduces the stiffness matrix calculation of isotropic and transversely isotropic materials.

Isotropic materials are equally elastic in all directions and anisotropic materials are stiffer in some directions than others. On the contrary, orthotropic materials have three different stiffnesses in three orthogonal directions. The familiar isotropic and transversely isotropic materials are generalized to materials with three stiffness values in three orthogonal directions by orthotropic materials. (Li & Barbič 2015)

The 6 x 6 symmetric elasticity tensor δ relates strain ε to stress σ via (Li & Barbič 2015):

$$\varepsilon = \delta\sigma \quad (25)$$

where

$$\varepsilon = \left[\varepsilon_{xx}\ \varepsilon_{yy}\ \varepsilon_{zz}\ 2\varepsilon_{xy}\ 2\varepsilon_{yz}\ 2\varepsilon_{zx}\right]^T$$

$$\sigma = \left[\sigma_{xx}\ \sigma_{yy}\ \sigma_{zz}\ \sigma_{xy}\ \sigma_{yz}\ \sigma_{zx}\right]^T$$

The inverse elasticity tensor δ^{-1} known as the stiffness matrix relates σ to ε via (Li & Barbič 2015):

$$\sigma = \delta^{-1}\varepsilon \tag{26}$$

Orthotropic materials have one Young's modulus, E_{xx}, E_{yy} and E_{zz}, for each orthogonal direction. They also have six Poisson's ratios, of which three are independent. On the contrary, isotropic materials are described by a single Young's Modulus and Poisson's ratio. Young's modulus E_{ii} describes the stiffness of the material when loaded in the orthogonal direction i. Poisson's ratio v_{ij} defines the contraction in direction j when an extension is applied to direction i. In a general anisotropic material, both the normal and shear components of strain are coupled with the normal and shear components of stress. For orthotropic materials, however, the normal and shear components are decoupled. Thus, shear stresses only have an effect on shear strains and normal stresses only have an effect on normal strains. Furthermore, individual shear stresses in the xy, yz and zx planes are decoupled from each other. In other words, strain ε_{ij} $(i \neq j)$ only depends on stress σ_{ij} via the shear modulus G_{ij}, where i and j denote directions. These assumptions cause the elasticity tensor to have 9 free variables. It can be written in the following block-diagonal form (Li & Barbič 2015):

$$\delta_{ortho} = \begin{bmatrix} \frac{1}{E_{xx}} & -\frac{v_{yx}}{E_{yy}} & -\frac{v_{zx}}{E_{zz}} & 0 & 0 & 0 \\ -\frac{v_{xy}}{E_{xx}} & \frac{1}{E_{yy}} & -\frac{v_{zy}}{E_{zz}} & 0 & 0 & 0 \\ -\frac{v_{xz}}{E_{xx}} & -\frac{v_{yz}}{E_{yy}} & \frac{1}{E_{zz}} & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{G_{xy}} & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{G_{yz}} & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1}{G_{zx}} \end{bmatrix} \tag{27}$$

The inverse of the orthotropic elasticity tensor is then (Li & Barbič 2015):

$$\delta^{-1} = \begin{bmatrix} A & 0 \\ 0 & B \end{bmatrix} \tag{28}$$

where

$$A = C\begin{bmatrix} E_{xx}(1-v_{yz}v_{zy}) & E_{yy}(v_{xy}-v_{zy}v_{xz}) & E_{zz}(v_{xz}-v_{xy}v_{yz}) \\ E_{xx}(v_{yx}+v_{zx}v_{yz}) & E_{yy}(1-v_{xz}v_{zx}) & E_{zz}(v_{yz}-v_{yx}v_{xz}) \\ E_{xx}(v_{zx}+v_{yx}v_{zy}) & E_{yy}(v_{zy}-v_{xy}v_{zx}) & E_{zz}(1-v_{xy}v_{yx}) \end{bmatrix}$$

$$B = \begin{bmatrix} G_{xy} & 0 & 0 \\ 0 & G_{yz} & 0 \\ 0 & 0 & G_{zx} \end{bmatrix}$$

$$C = \frac{1}{1-v_{xy}v_{yx}-v_{yz}v_{zy}-2v_{yx}v_{zy}v_{xz}}$$

The hysteretic damping matrix $i[D]$ included in equations (14) and (24) can be formed by multiplying each Young's moduli E and shear moduli G in δ^{-1} with the imaginary unit i and the respective damping loss factor ɳ. The damping loss factor can vary in each direction. (Roivainen 2009, p. 78) The stiffness and hysteretic damping matrices have to be defined for each material present in the finite element method (FEM) model separately.

Two of the three orthogonal directions are equally stiff for a transversely isotropic material. For such a material a plane exists in which it is isotropic, but in the orthogonal direction it is not. In that case $E_{xx} = E_{yy}$, $v_{xy} = v_{yx}$, $v_{xz} = v_{yz}$, $v_{zx} = v_{zy}$, $G_{xy} = E_{xx}/2(1+v_{xy})$ and to keep the elasticity tensor symmetric, the Poisson's ratios have to satisfy the following condition (Li & Barbič 2015):

$$\frac{v_{ij}}{E_{ii}} = \frac{v_{ji}}{E_{jj}} \tag{29}$$

for all $i \neq j$, where i and j denote directions. A further simplification is the isotropic material which has only two free parameters E and v. In that case we have $E_{xx} = E_{yy} = E_{zz} = E$, $v_{ij} = v$ for all i, j and $G_{xy} = G_{yz} = G_{zx} = E/2(1+v)$. (Li & Barbič 2015)

Because the elastic strain energy of the orthotropic material has to be a positive-definite function of ε, the inverse elasticity tensor δ^{-1} must be positive-definite. Thus the following conditions have to be satisfied: $G_{xy} > 0, G_{yz} > 0, G_{zx} > 0, {v_{xy}}^2 < \frac{E_{xx}}{E_{yy}}, {v_{yz}}^2 < \frac{E_{yy}}{E_{zz}}$ and ${v_{zx}}^2 < \frac{E_{zz}}{E_{xx}}$. In the isotropic case it is well-known that only the following conditions have to be satisfied: $-1 < v < \frac{1}{2}$ and $E > 0$. (Li & Barbič 2015)

2.2.5 Substructuring with component mode synthesis

The size of the analysis problem is often a cause of concern for the analyst especially in case of transient analyses. The development of a rotating electrical machine is a multidisciplinary task, and many of its components are usually designed and dynamically

analyzed by different departments. This often means that the dynamic analysis results carried out by different departments can not directly be utilized later when the whole rotor-bearing-casing structure is analyzed at once. (Shanmugam & Padmanabhan 2006, p. 1)

The component mode synthesis (CMS) is a good method for solving the above mentioned challenges. CMS enables analyzing the individual components separately and the problem size to be reduced for each component. Finally, a complete model of highly reduced size can be built and analyzed to obtain the results. Furthermore, the benefit of CMS is that the analyst can perform design modifications for individual components to achieve the wanted dynamic properties without having to analyze all other components of the complete system again each time. Only the particular modified component has to be analyzed again, and the previous analysis results of other components can be assembled to assess the behavior of the complete system. The result is reduction of the number of design iterations. (Shanmugam & Padmanabhan 2006, p. 1-2)

Particularly in rotordynamics, it can be very beneficial to analyze separately the rotor and the stator of the machine and then connect them together at the bearing points. This way of reducing the model size is ideally suited for rotordynamic analysis because rotating machinery is usually made of parts connected with each other in a very limited number of locations, such as the bearings or dampers. (Genta 2005, p. 197)

One method of component mode synthesis is the fixed interface method, also known as the Craig-Bampton method. For example, consider a structure shown in Figure 12 to be divided into two components. These components are discretized by the FEM. Both components have internal nodes independent of other components and boundary nodes connecting to other components. The corresponding DOFs of each component are called internal DOFs and boundary DOFs, respectively. (Qu 2010, p. 321-322) Each subsystem is analyzed separately in order to acquire the component modes of each subsystem. Component modes include vibration modes with the substructure interface DOFs fixed and constraint modes which are static displacement patterns produced by applying a unit displacement to each interface DOF after each other while keeping other DOFs fixed. Thus, the system is reduced by retaining a subset of modes for each subsystem and by the number of interface DOFs between the systems. The complete reduced system is presented using a combination of physical and modal coordinates and it is built by assembling the substructures along shared interface nodes. The complete reduced system contains the full system dynamics. (Matthew *et al.* 2010, p. 11) More information and a detailed mathematical formulation of CMS is presented in Qu (2010).

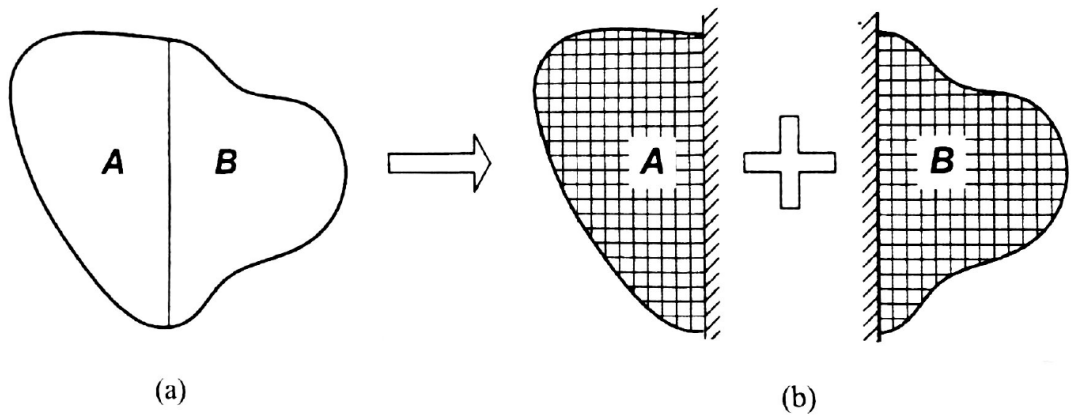

Figure 12. Division of one discretized structure into two components using a fixed interface: (a) the complete structure; (b) the components. (Qu 2010, p. 322)

The reduced system of a non-rotating rotor can be created with *Elmer*. After creating the reduced system, dampers or rotating disks can be connected to the interface DOFs to take into account the effects of bearings or rotation. This way the amount of DOFs of the complete system is greatly reduced, which allows solving the Campbell diagram of the complete system efficiently with a reduced amount of computational effort. This can be accomplished with a software such as *Modysol* developed at VTT, Technical Research Centre of Finland. (Klinge 2005)

2.3 Specification of the model generator

This chapter specifies the model generator in subchapters 2.3.2…2.3.5. The specification includes the connection to the automated computation system, the non-functional and functional requirements, and the rotor structure and its variants to be modeled. Moreover, the software tools utilized in this study are presented in subchapter 2.3.1.

2.3.1 Descriptions of the software tools

This subchapter shortly presents the software tools utilized in this thesis. These tools were chosen because they are cost free, highly customizable and allow automating processes.

Salome is an open-source pre- and post-processing software. The geometry module of *Salome* (GEOM) provides a rich set of functionalities to create, modify, or import complex CAD models. The geometric shapes may be designed either using the graphical user interface (GUI) or the TUI through Python scripts. Each functionality of the GEOM module is available in the TUI, which allows parametric model creation. The geometry kernel of the GEOM module is based on the Open CASCADE Technology which creates a boundary representation (BREP) and maintains the topological structure required by the subsequent meshing operations. The mesh module of *Salome* (SMESH) provides a large

range of meshing algorithms. A mesh can be divided in groups to distinguish between different regions of the geometry. This allows differentiation between mesh properties or mesh types. Groups also allow the definition of local boundary and initial conditions. The groups can be automatically obtained from the geometry model. Moreover, the meshing process can be entirely described by Python scripts in the TUI to handle complex studies or to ensure full reproducibility or parametrization of the simulation workflow. (Salome 2014)

Adept is a software in use at ABB which acts as a user interface to electrical machine data and calculation tools. No matter which calculation engine is utilized, the inputs and results are handled in *Adept* in the same way. (Ryyppö 2015, p. 6) The data of one product can be exported from *Adept* as an adept.dat file which contains the values and variable names separated with semicolons. Among this data are the geometric feature parameters and dimensions which can be exploited in *Salome* for geometry creation.

Elmer is an open-source finite element software for multiphysical problems which is developed by CSC, the Finnish IT Center for Science. Among others, it includes physical models for structural mechanics and numerical methods for solving dynamic analyses, such as eigenvalue analyses. It also supports mesh partitioning by *Metis* for parallel solving. *ElmerSolver* is utilized for assembly and solution of the finite element equation. *ElmerPost* can be utilized for post-processing the results. (Zwinger 2008)

Octave is a high-level language intended mainly for numerical computations. The language is mostly compatible with *Matlab* and it is easily expandable and customizable through user-defined functions written in the own language of *Octave*. (Octave 2015)

2.3.2 Connection to the automated computation system

Figure 13 shows the general workflow of a computation process of which one phase is the mesh generation with *Salome*. The three possible analysis types which were decided to be automated are:

1. Eigenvalues of a non-rotating free rotor
2. Eigenvalues of a non-rotating rotor on bearings
3. Eigenvalues of a rotating rotor on bearings

To start the process, the user defines and uploads the necessary files to the computation server and executes the computation process. A UNIX shell script job.sh is created based on the information in the uploaded files. It is then added to a work queue of a portable batch system (PBS) which is applied for allocating resources among computational tasks. The general computation process parameters given by the user include the requested analysis type, the number of modes to be solved, the rotational speed, the requested

number of processor cores, the requested amount of memory and the requested computation time. Once there are enough processor cores and memory available for the requested computational task, the job.sh script is executed. It initiates the mesh generation with *Salome* and then transforms the obtained mesh to *Elmer's* native ElmerGrid format. If the user has defined an analysis type which requires adding boundary conditions to bearing nodes, the bearing nodes are fixed by manipulating the mesh files and the solver input file. After this, the mesh is partitioned for parallel computation to correspond to the amount of cores requested by the user. In the end, the partial results are fused to obtain the full solution. Parallel computation is applied to reduce the required computation time. Before the solver can be called, the stiffness matrices for all materials present in the model are calculated using *Octave* and the formulas described in subchapter 2.2.4. *Octave* outputs a text file containing the stiffness matrices for all materials present in the model with or without hysteretic damping included depending on the case. The created text file is then included in the solver input file of *Elmer*. The solver input file of *Elmer* is written using Python based on the user-defined computation parameters. Finally, the solver is executed and the results are forwarded to the user after the solution is obtained. As a conclusion, *Salome* is executed only once during one computational task. It delivers the mesh and information for utilization in later phases of the process.

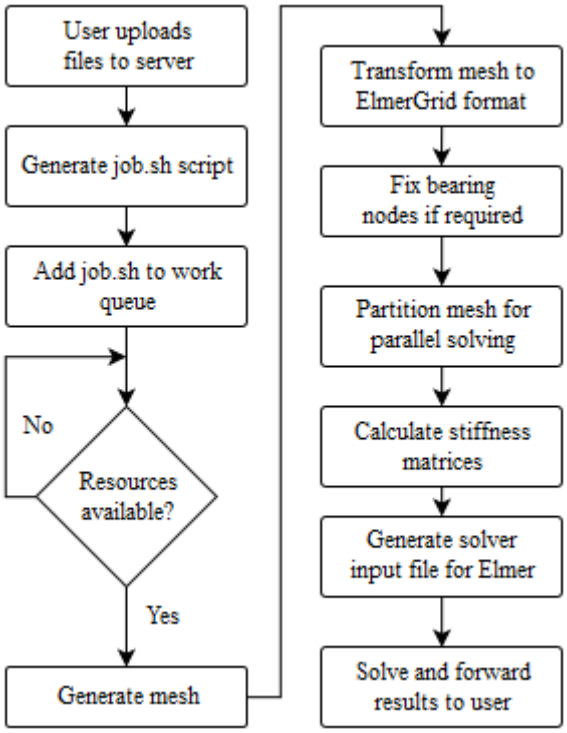

Figure 13. The general workflow of one computation task.

2.3.3 Functional requirements

A functional requirement defines a function of a system and its components. This thesis defines a function as inputs, behavior, and output. The functional requirements of the model generator are discussed in this subchapter.

Figure 14 shows the desired inputs and outputs of the main model generator program in *Salome*. The main program should obtain files named adept.dat, rotor_data.gdc and parameters.txt as inputs. All inputs and outputs of the system should be defined in SI-units. The adept.dat file contains the necessary dimensions and geometry information to create geometry for analysis purposes. Ideally this file would contain also the standard node designations and shaft dimensions to assign nodes to standardized positions for defining boundary conditions, interfaces, and analyzing nodal results. However, at this point of the project, adept.dat contains only a limited amount of information about the shaft geometry and standard nodes. Therefore, a separate rotor_data.gdc file should be utilized to describe the complete shaft geometry and standard node locations. The file parameters.txt should contain all of the user-defined analysis, model creation, and default meshing parameters.

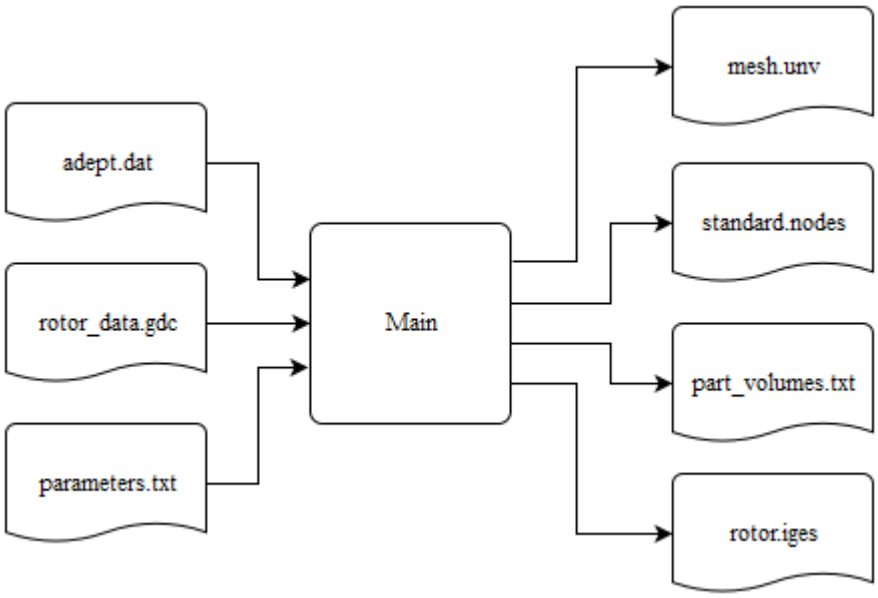

Figure 14. The main program inputs and outputs.

The model generator output should be a mesh file in .unv file format because it enables inclusion of element volume group data of different bodies and it is compatible with *Elmer*. The volume group information is important to assign the correct mechanical material properties to correct parts. Additionally, information about the locations and node numbers of standard nodes should be provided for later node mapping in the solving and post-processing phases in the standard.nodes file. The created geometry and part

volumes should be exported as rotor.iges and part_volumes.txt to enable geometry validation and debugging once the system is tested in a wider scope.

2.3.4 Non-functional requirements

Non-functional requirements are also known as quality requirements. They describe, for example, the reliability and performance. This subchapter discusses the non-functional requirements of the model generator.

The model generator does not generate exact geometry ready for creating *e.g.* manufacturing drawings. It creates geometry based on dimensions for engineering analyses which means simplified geometry when compared to rotating electrical machines in reality. Thus, the CAE-centric approach should be applied. Even these simplified geometries may be altered slightly based on simplification rules to achieve better results in the subsequent meshing phase.

The model generator shall be as fast and stable as possible to create mesh files. However, the most important factors are stability and reliability, and therefore the initial meshing parameters should be selected and adjusted conservatively. The higher the amount of nodes, the longer is the time required for mesh generation. Thus, minimizing the amount of nodes by using coarse meshing parameters can be beneficial. However, trying to mesh too complex geometry with too coarse meshing parameters causes the meshing process to fail. After a failed mesh computation attempt, the whole meshing process has to be restarted.

The meshing parameters should be adjusted automatically if the meshing process fails due to trying to generate a too coarse mesh. This ensures that the mesh can be generated and the whole computation process explained in subchapter 2.3.2 can be completed. The rotor mesh should preferably not have more than 300 000 nodes, which means 900 000 DOFs, as this can be considered a reasonable amount for fast solving.

2.3.5 The rotor structure and its variants

This subchapter discusses the rotor variants which were chosen to be included in the model generator and the rotor structure itself in more detail. The main rotor type to be modeled was chosen to be one of an induction motor shown in Figure 15. This rotor was chosen because there were experimental measurement data and reference 3D models available for comparisons and validations. The figure also shows the main parts of the rotors which were decided to be included in the model generator.

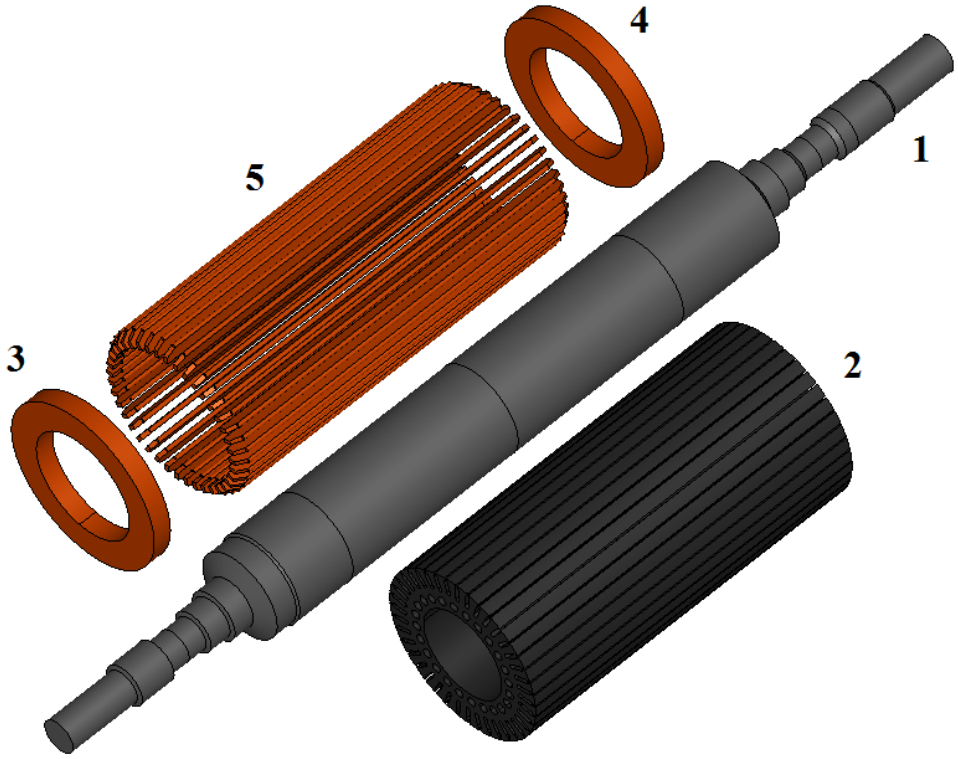

Figure 15. The rotor structure. The parts are: 1. Shaft, 2. Core, 3. Short-circuit ring 1, 4. Short-circuit ring 2, 5. Bars.

Figure 16 shows the two slot types which were chosen to be included in the model generator. Each slot type has a corresponding bar type which fits in the slot of the rotor core.

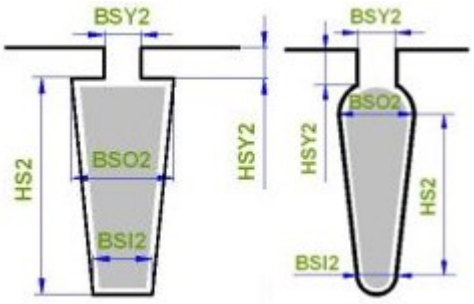

Figure 16. Rotor core slot type 1 and slot type 2 and their dimensions. The grey areas show the respective bar geometries. (Figure from Adept)

From the possible rotor core axial cooling duct types described by the parameters VAC2, type 0, 1, and 2 shown in Figure 17 were chosen to be included in the model generator. For VAC2 type 1, the parameter NROW2 defines the number of cooling duct rows.

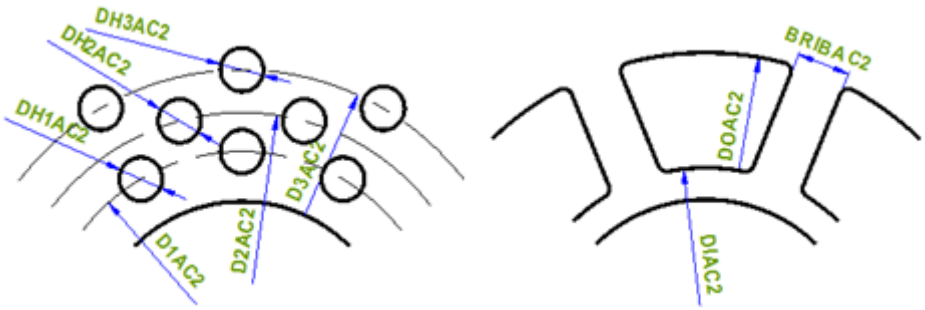

Figure 17. VAC2 Type 1 and 2 of rotor core axial cooling ducts from left to right. Type 0 means no cooling ducts. NROW2 defines the number of cooling duct rows for VAC2 Type 1. (Figure from Adept)

There are many details in the rotor which were omitted to keep the meshing and geometry generation process simple. For example, the possible rotor core radial cooling ducts were omitted as well as cooling fans and small details in the shaft. The effect of radial cooling ducts on the axial Young's modulus should be taken into account in the material parameter calculation by calculating an effective Young's modulus. Inclusion of the radial ducts would have made the automated geometry generation and meshing unnecessarily complex.

Thus, there are the following feature variables causing different rotor variants:

- BSO2 shown in Figure 16 can be equal or inequal to the dimension BSI2
- the possibility to model the bar with or without side contact to the slot side surfaces
- three possible axial cooling duct types
- the number of cooling duct rows for type 1 cooling

If the varying dimensions are omitted, only combining different features in the above list allows the generation of 40 feasible rotor variations. When all the dimensions, number of cooling duct holes, number of bars, and other parameters affecting the geometry generation are considered, the amount of feasible variations increases significantly. There are 20-50 dimensions which can be varied depending on the case.

2.4 Creation of the model generator

This chapter explains the creation of the model generator. The decisions made during the development process and the methods applied are discussed in more detail. The subchapter 2.4.1 discusses how the geometry is generated and simplified. The subchapter 2.4.2 discusses the mesh generation and assignment of boundary conditions in more detail. Finally, subchapter 2.4.3 discusses testing a method of using an extruded mesh to reduce the amount of DOFs in the model.

2.4.1 Geometry generation

Although *Salome* allows modeling mesh directly, modeling geometry first and then meshing it was chosen due to the more diverse functionality available in the GEOM module. In some cases it might be beneficial to directly model mesh in the SMESH module and skip the geometry modeling phase. During testing of moving the generated mesh from *Salome* to *Elmer*, it was noted that *Elmer* recognizes element volume groups based on the order they were created in *Salome*. This is also the order in which they appear in the created .unv mesh file. A default order for parts and volume group creation was established for consistency. This order was also decided to be the element volume group numbering when generating the solver input file for *Elmer* to assign mechanical material properties:

1. Shaft
2. Rotor core
3. Short-circuit ring 1
4. Short-circuit ring 2
5. Bars

The inputs and outputs of the main program were illustrated in Figure 14 in the subchapter 2.3.3. The source code of the main program is shown in "Appendix 1: The model generator main program". The main program is executed inside *Salome*. It reads in the parameters from adept.dat and parameters.txt and initializes the model creation process. It calls for sub-functions which create the individual parts and return them to the main program. The parts are then moved to the correct positions. Parts of which more than one piece is required are copied to meet the required amount defined by adept.dat parameters. The inputs and outputs of the sub-functions of the main program are discussed in the subsequent paragraphs.

The inputs and outputs of the Make_Shaft sub-function are shown in Figure 18. It creates and returns the shaft geometry based on the rotor_data.gdc file. The shaft centerline is on the negative side of the z-axis because it is a common convention with rotating electrical machines. There are lists named VNODES and SNODES inside the function. They are

visualization nodes and standard node designation identifiers (IDs), respectively. Visualization nodes should later be utilized for creating visualizations of the mode shapes of rotors to be compared with reports generated with *Ardas*. The standard node designation list exists to check if a node designation in rotor_data.gdc is a standard node. The Make_Shaft sub-function returns a list of node numbers and coordinates of standard and visualization nodes among a list of vertices created to these points. The visualization node points are distributed evenly along the shaft centerline. The lists Standard_nodes and Vertices are later utilized to ensure that there are nodes located on the desired points points along the shaft centerline in the meshing phase. Because the actual node numbers which are created during meshing are unknown in the geometry generation phase, standardized numbers are utilized and the actual node numbers are obtained after meshing. The Make_Shaft sub-function also returns Rcore_Center, the z-coordinate of the rotor core center point on the shaft centerline to enable correct positioning of other rotor parts.

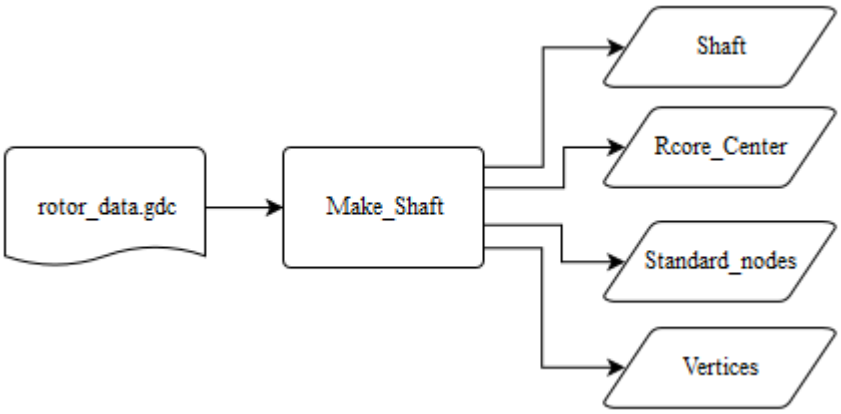

Figure 18. Inputs and outputs of the Make_Shaft sub-function.

The rotor_data.gdc file contains a table of topology information and shaft dimensions which are adopted and modified from values utilized in *Ardas*, a rotordynamic analysis software (ALSTOM 2002). Table 1 shows the possible values at each column and their explanations. Only the values employed in this thesis are shown, although there are other possible values in the rotor_data.gdc file.

Table 1: Columns of the rotor_data.gdc file and their explanations in the info columns. A, B and C denote dimensions in meters.

Topology	Info	NEL	Info	ELEN	Info	DOL	Info	DOR	Info
0	Plain shaft	1001	Bearing center D-end	A	Segment length	B	Diameter in the left end of the segment	C	Diameter in the right end of the segment
1000-1999	Coupling	1002	Bearing center N-end						
2000-2999	Bearing	1003	Rotor package center						
3000-3999	Core	1004	Shaft D-end						

The values are separated by commas and each row represents one segment of the shaft. The whole shaft is compiled of these small segments. The left end of the segment is always the corresponding standard node location. The element identification number (NEL) used to identify the segment can be any number, but to assign for example a bearing node to a desired position, the corresponding standard node designation ID value should be applied. For example, the following two rows in rotor_data.gdc:

2000, 1, 0.1, 0.2, 0.2
2001, 1001, 0.1, 0.2, 0.2

would create two shaft segments with the length of 0.1 m, diameter 0.2 m and there would be a bearing center point in the middle of these two segments. Both segments would belong to the topology group “Bearing”.

The sub-functions Make_Rcore, Make_Sring and Make_Bar shown in Figure 19 each create the geometry of one respective part and return it to the main program. The geometries are mainly created by calculating locations of geometry points, creating vertices to those locations and then creating sketches and solids based on them. There are three modeling parameters in parameters.txt named BAR_SCONTACT, GAP and BAR_RTOL. The parameter BAR_SCONTACT is 0 if the bars are modeled to have no side contact with the rotor core and 1 when side contact is desired. Usually the value should be 0. This is due to the manufacturing method, in which the bars are swaged when they are assembled to the rotor core slots causing them to have no side contact (Kanninen 2006). If the value is 0, the side lines of bars are modeled using arcs, and if the value is 1, the cross-section of bars are modeled in the default form as in Figure 16. If BAR_SCONTACT is set to 0, GAP is a multiplier which defines the relative reduction

of the bar width at the side line midpoint with respect to the case with side contact and full bar width.

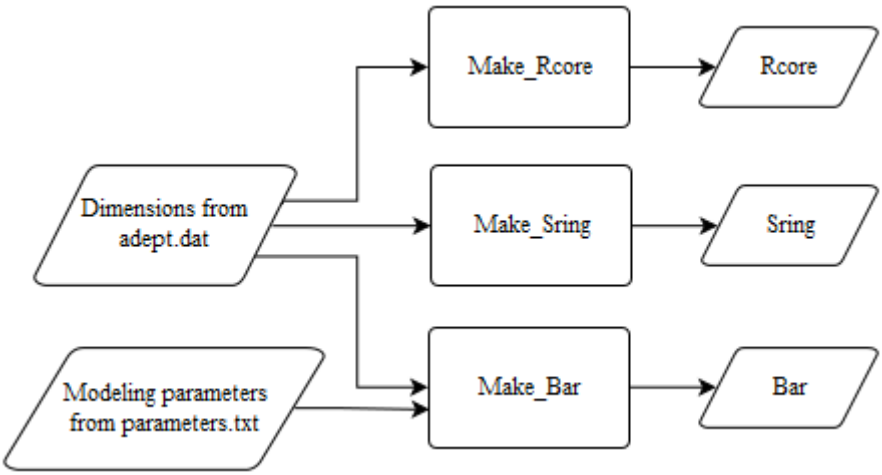

Figure 19. Inputs and outputs of the Make_Rcore, Make_Sring and Make_Bar sub-functions.

Figure 20 shows the upper and lower surfaces of bars for slot type 2 outside the rotor core, which caused problems in the later meshing phase. Meshing succeeded only with very fine meshing parameter values causing the mesh to be unnecessarily dense.

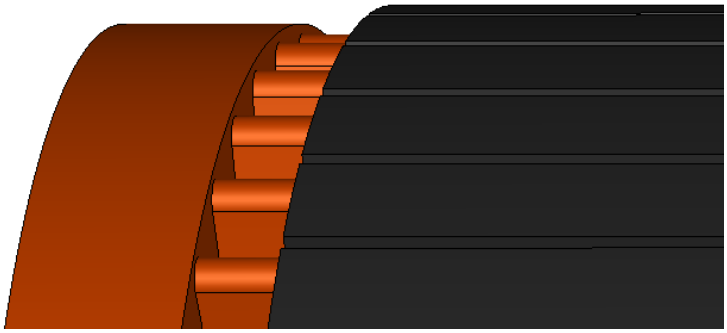

Figure 20. The small and round bar tip surfaces and a small distance between bar tips and the short-circuit ring top surface cause the mesh to be unnecessarily dense and therefore they have to be removed.

In order to avoid this problem some geometry simplification functionality was added to the Make_Bar function. The unnecessary small and round faces were cut out of all rotors with slot type 2 as can be seen in Figure 21. The bar tip radius from the origin to the left corner of the bar cross-section is calculated subsequently. The same value is also calculated directly in the case of slot type 1 bars, without any preceding modifications to the bar geometry. After this, a maximum tolerated bar tip radius is defined by multiplying the short-circuit ring top surface radius by a tolerance factor BAR_RTOL. The tolerance factor is defined to be between 0 and 1.

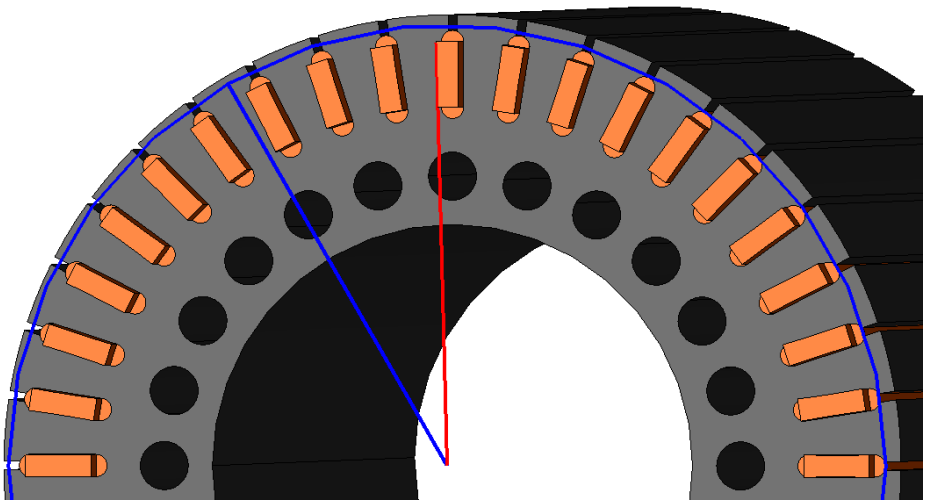

Figure 21. The slot type 2 bars after cutting out the unnecessary round faces. The blue line represents the short-circuit ring radius and the red line represents the bar tip radius. If the bar tip radius exceeds the short-circuit ring radius multiplied with a tolerance factor, the bars are chamfered.

If the bar tip radius exceeds the maximum tolerated bar tip radius, the bars are chamfered so that the bar tip radius at the end of the bar is below the tolerated value, as shown in Figure 22. These modifications make the automated meshing procedure faster and more reliable while producing a less dense mesh. However, these modifications cause a small reduction in the total volume of the bars which may lead to a small inaccuracy in the simulation results later.

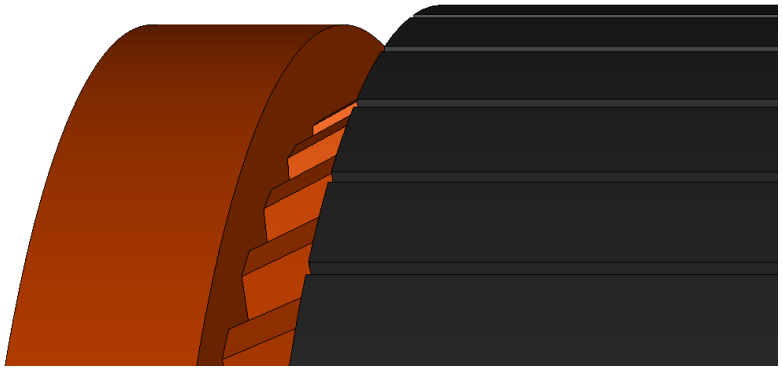

Figure 22. The round bar tip surfaces cut out and the portion of the bar outside of the rotor core chamfered to avoid the small distance between the short-circuit ring top surface and the bar tip.

2.4.2 Mesh generation and assignment of boundary conditions

Utilizing SI-units and small values caused problems in the meshing phase, since *Salome* rounds the values if they have many decimals. This was avoided by creating a small sub-function under the main program which scales all the dimensions with a scale factor S before generating geometry and meshing. After creating the parts, the main program calls

for the sub-function Create_Mesh, whose inputs and outputs are shown in Figure 23. It takes Partition_1 created in the main program as an input. Partition_1 is the complete assembled rotor geometry with duplicate faces between parts removed to enable conformal meshing. It takes also the part geometries Shaft, Rcore, Sring_1, Sring_2 and Bars as inputs to create element volume groups based on geometry after the meshing is completed. Furthermore, it takes the list Standard_nodes and default meshing parameters from parameters.txt as inputs. It returns the created mesh with element volume groups for each part and the list Standard_nodes supplemented with real mesh node IDs obtained from the generated mesh. The created mesh is later scaled back to SI-units using the inverse of the scale factor S. This ensures that the simulation results are always in SI-units, independent of the scale factor S.

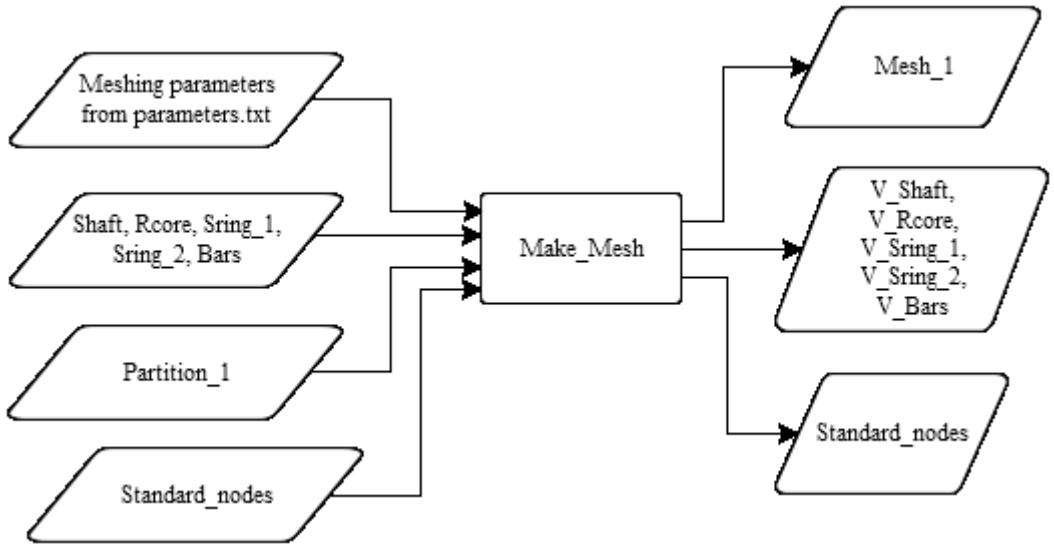

Figure 23. The inputs and outputs of the Make_Mesh sub-function.

Since the meshing had to be completely automated for multiple rotor variations without any user influence, the most reliable meshing algorithm had to be chosen. The choice was *NETGEN*, an automated cost-free 3D tetrahedral mesh generator. There are also numerous other meshing algorithms available, but they require more input to be able to create the mesh. Usually they require algorithms to be defined separately for one-dimensional (1D), 2D and 3D discretization while *NETGEN* works directly for 1D, 2D and 3D with the same algorithm (Salome 2015). There are also functions inside *Salome* to take advantage of symmetry in the meshing phase (Salome 2015). These were not exploited due to the high number of rotor variants which would add unnecessary complexity in conformal meshing and topology management.

NETGEN could mesh complex 3D geometry reliably, and the only parameters which required adjustment in case of meshing failure were observed to be (Salome 2015):

- Max Size - the maximum linear dimensions for mesh cells
- Min Size - the minimum linear dimensions for mesh cells
- Growth Rate - the maximum relative difference of linear dimensions of two adjacent cells (e.g. 0.3 means 30%)
- Nb. Segs per Edge - the minimum number of mesh segments in which edges are split
- Nb. Segs per Radius - defines the size of mesh segments and mesh faces in which curved edges and surfaces are split

The other meshing parameters are kept constant to keep the number of variables as small as possible in the current development phase. These parameters are (Salome 2015):

- Second Order - if this option is on, second order nodes are created.
- Fineness - the level of meshing detail. It can be very coarse, moderate, very fine, or custom.
- Allow Quadrangles - if this option is on, quadrangle elements are allowed.
- Optimize - if this option is on, the initially created mesh is modified in order to improve quality of elements. The optimization process is rather time consuming comparing to creation of the initial mesh.
- Fuse Coincident Nodes on Edges and Vertices - allows merging mesh nodes on vertices and edges which are geometrically coincident but topologically different.
- Limit Size by Surface Curvature - if this option is on, the size of mesh segments and mesh faces on curved edges and surfaces is defined using value of Nb. Segs per Radius parameter, and number of segments on straight edges is defined by the value of Nb. Segs per Edge parameter. If this option is off, then the size of elements is defined by three parameters only: Max Size, Min Size, and Growth rate.

Second order was set off, since the mesh can easily be transformed from linear elements to second order elements in *Elmer*. Fineness was set to custom to enable adjusting Growth rate, Nb. Segs per Edge and Nb. Segs per Radius. Using quadrangles could make the mesh better structured, but during testing it caused instability. Therefore Allow Quadrangles was set off to increase mesh generation reliability. The Fuse Coincident Nodes and Limit Size by Surface Curvature parameters were defined to be on.

Figure 24 shows the functioning of the Make_Mesh function. A while loop was created to check if the mesh computation succeeded. Most commonly the mesh computation failed because the Min Size or Growth Rate values were too great. The first time the mesh computation fails, the program adjusts the mesh Max Size, Min Size, and Growth rate parameters to smaller values and the Nb. Seg per Edge and Nb. Seg per Radius to greater values and tries to compute the mesh again. If the second mesh computation attempt fails, only the values of the Max Size, Min Size, and Growth Rate parameters are reduced until the mesh computation is successful or the maximum amount of retries is reached. If the maximum amount of retries is reached, the whole process terminates. Trying to compute the mesh over and over again costs time and therefore the multipliers to reduce the Max Size, Min Size, and Growth Rate parameters are defined to be 0.90, 0.75, and 0.90, respectively. The adjustments are rough, but they ensure that at maximum only a few attempts are required.

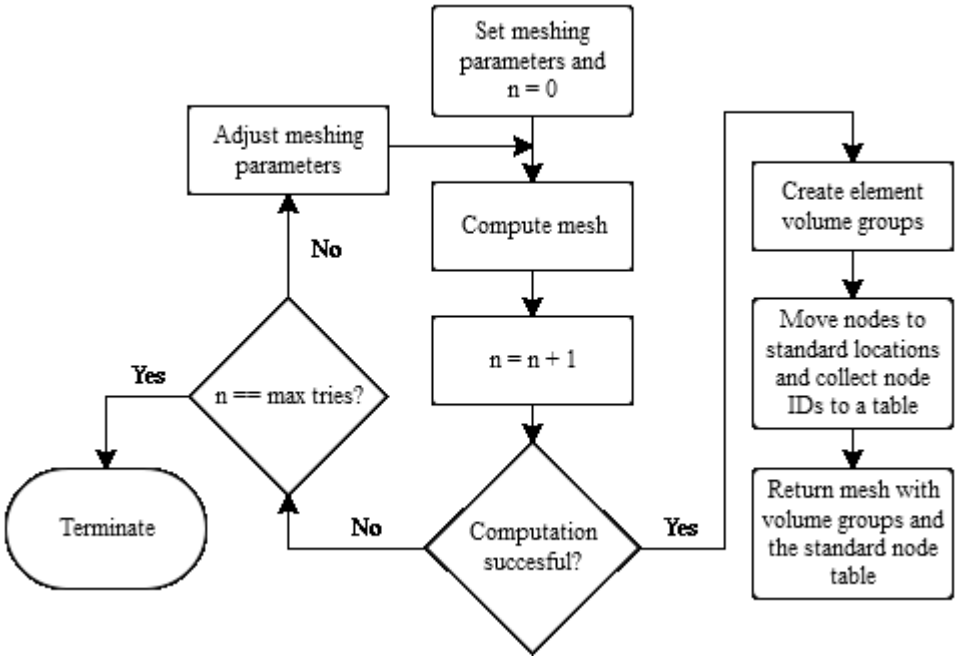

Figure 24. The functioning of the Create_Mesh sub-function.

The default values for modeling and meshing shown in Table 2 were chosen by testing the meshing of 24 different test cases in "Appendix 2: Test cases and benchmarking" and choosing meshing parameters which produce as rough meshes as possible without more than 2 meshing failures. The test cases were chosen to represent a wide variety in both the geometry features and dimensions. The meshing of test cases 10 and 22 using the defined default parameters failed on the first attempt but succeeded after the first adjustment loop. Meshing test case 6 rotor succeeded after the second adjustment loop. In the ideal case the adjustment loop would never be required, but it would cost more time to define the default meshing parameters so that an unnecessarily dense mesh is generated

for approximately 90% of the rotors. On average the model generation and meshing for the test cases took 3 minutes and 35 seconds.

Table 2: Default modeling and meshing parameter values.

Parameter	Value
S	1000.0
BAR_RTOL	0.93
GAP	0.03
MaxSize	0.25
MinSize	0.013
SecondOrder	0
Optimize	1
UseSurfaceCurvature	1
FuseEdges	1
GrowthRate	0.07
SegsPerEdge	0.5
SegsPerRadius	1.5

After the mesh has been computed, the element volume groups are created by using the built-in SMESH module functions smesh.GetFilter and smesh.GetInPlace of *Salome*. *Salome* finds the elements in the mesh based on the geometry. For example, the following two rows:

```
filter = smesh.GetFilter(SMESH.VOLUME, SMESH.FT_BelongToGeom,
geompy.GetInPlace(Partition_1, Shaft))
V_Shaft = smesh.GetFilter(SMESH.VOLUME, 'V_Shaft', filter)
```

create an element volume group called V_Shaft based on the geometry object "Shaft" inside Partition_1. Exporting the mesh.unv file required a very long time when the above volume group creation method was used without the geompy.GetInPlace –function. This would have slowed the model generation process down considerably. Element volume groups are created for each part to be able to assign the correct mechanical properties for the elements later before the analysis process.

A built-in *Salome* SMESH function MeshToPassThroughAPoint(x, y, z) was exploited to double check that nodes actually are in the designated positions and to get their node ID numbers from the generated mesh. This was done because it is not possible to directly force *NETGEN* to create nodes to certain points. However, it was noticed that nodes are often created to vertices, and therefore vertices were created to standard node locations before meshing. Thus, the MeshToPassThroughAPoint(x, y, z) function finds and moves the closest node to the given coordinates and returns the node ID number. If there already is a node in the given location, the function only returns the node ID number and does not move the node. This is highly likely due to the vertices created to these locations. The

acquired mesh node ID numbers and their coordinates are then written to the Standard_nodes list which is later exported to a file standard.nodes to define boundary conditions for the solving process with *Elmer*.

The format of the columns of standard.nodes is NEL, NodeID, x, y, z. The number of rows equals the number of standard and visualization nodes in the model. Visualization nodes are created to enable visualizing the mode shapes in the same format as *Ardas* for validation. The contents of the standard.nodes file looks as follows:

1004 28704 0.0 0.0 0.0
2001 57590 0.0 0.0 -0.270
1001 57496 0.0 0.0 -0.360
1003 54383 0.0 0.0 -1.275
1002 56865 0.0 0.0 -2.121

The boundary conditions of bearings are, in this development phase assigned to only two nodes, or more precisely to two point elements. In a more realistic set-up, they should be applied to the whole surface node groups which are inside the bearings. In this early development phase, it is considered to be an acceptable simplification to keep testing and development easier. When the mesh created with *Salome* is transformed to *Elmer* mesh format, *Elmer* creates a mesh.header and mesh.boundary file among other mesh files. To add two 101 *Elmer* point elements for defining boundary conditions, these two files have to be modified. A Python script which rewrites both files was created for this purpose. It opens the previously created standard.nodes file and reads in the bearing node numbers and creates the two point elements to these node locations based on node numbers. The two point elements for boundary conditions are created even though they would not be utilized, since they do not affect the solving process, unless boundary conditions are applied to them in the *Elmer* solver input file.

2.4.3 Utilization of an extruded mesh along the rotor core

The utilization of an extruded mesh along the rotor core allows reducing the total amount of DOFs. This should reduce the total meshing and solving time and therefore this method was also tested by creating a modified version of the main program and its sub-functions. A full tetrahedral mesh causes the mesh to be dense along the rotor core. To enable using an extruded mesh along the rotor core, the rotor has to be divided into 9 volume groups instead of 5 as required with a full tetrahedral mesh:

1. Shaft 1
2. Shaft 2
3. Shaft 3
4. Rotor core
5. Short-circuit ring 1
6. Short-circuit ring 2
7. Bars 1
8. Bars 2
9. Bars 3

A volume group named G_Extrusion has to be created under Partition_1 based on the parts Shaft 2, Rotor core, and Bars 2 to enable conformal meshing while utilizing an extrusion algorithm along the rotor core length. Shaft 2 and Bars 2 are the portions of the parts inside the rotor core. The rotor core face also has to be identified and discretized using tetrahedral elements as shown in Figure 25. Furthermore, the edges along the rotor core length have to be identified and discretized to a predefined amount of segments. During testing of the modified version of the model generator, it was noted that the edges along the rotor core length must be symmetrically aligned with respect to the y-axis. If this condition is not fulfilled, the mesh computation fails. Thus, if there is an edge on a surface which is coincident with another surface, the other surface also has to have an edge which is coincident with the before mentioned edge. This meant that for some rotor variants, additional edges had to be created to bar top surfaces for the edges of the core to have coincident pairing edges on the bars. In addition, some parts had to be rotated to align the edges correctly.

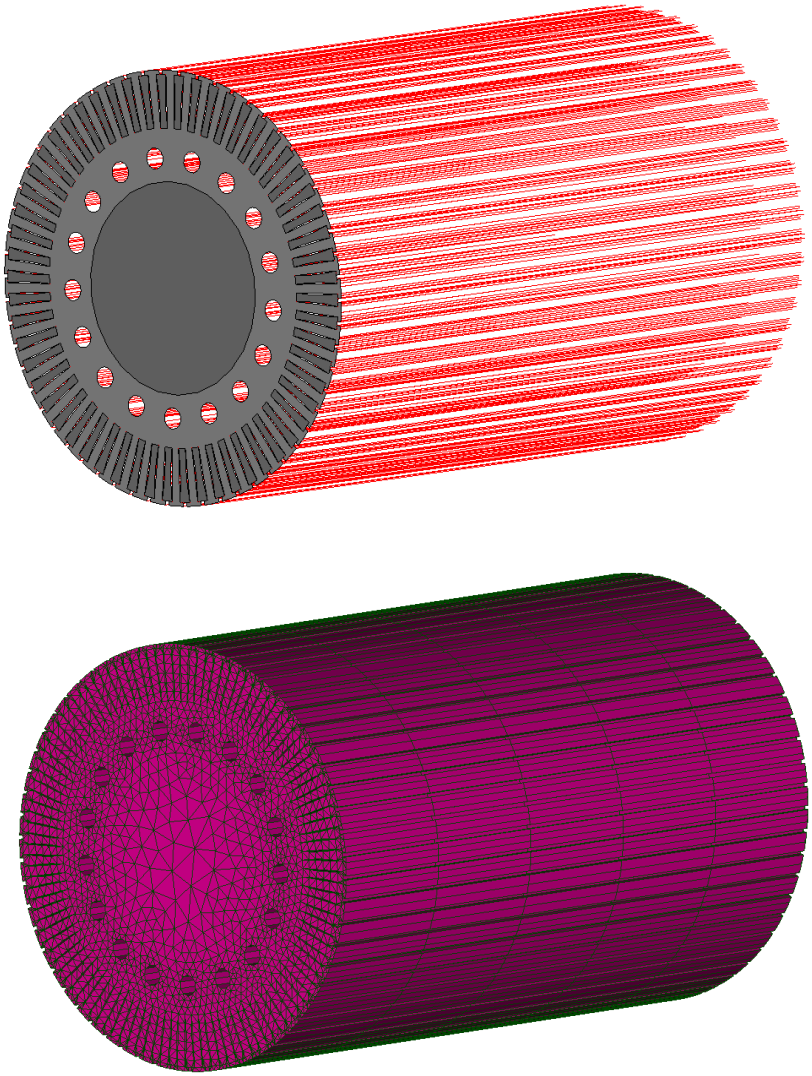

Figure 25. On the top: the rotor core face to be discretized with a tetrahedral 2D mesh and extruded along the core edges which are discretized to a number of segments. On the bottom: the 3D extruded mesh of the volume group G_Extrusion with 5 segments.

As a result, with one test rotor the full tetrahedral mesh had 80130 nodes and with the combined extruded and tetrahedral mesh it had 54882 nodes. The geometry and mesh were generated with the parameters shown in Table 3. This means 32% less DOFs even when the meshing parameters of the combined mesh were finer than those of the full tetrahedral mesh. However, the effects of a combined mesh on the simulation results should be studied. That was not part of this study. Both meshes were created with parameters which produce the roughest mesh without a failed mesh computation. The test rotor with the combined mesh is shown in Figure 26.

Table 3: Comparison of the amount of nodes generated with a full tetrahedral and a combined tetrahedral and extruded mesh for the same geometry. The given meshing and modeling parameters were applied.

Parameter	Full tetrahedral	Tetrahedral and extruded
S	1000.0	1000.0
BAR_SCONTACT	0	0
BAR_RTOL	0.93	0.93
GAP	0.03	0.03
MaxSize	0.25	0.25
MinSize	0.013	0.012
SecondOrder	0	0
Optimize	1	1
UseSurfaceCurvature	1	1
FuseEdges	1	1
GrowthRate	0.07	0.06
SegsPerEdge	0.5	3
SegsPerRadius	1.5	5
Number of nodes	80130	54882

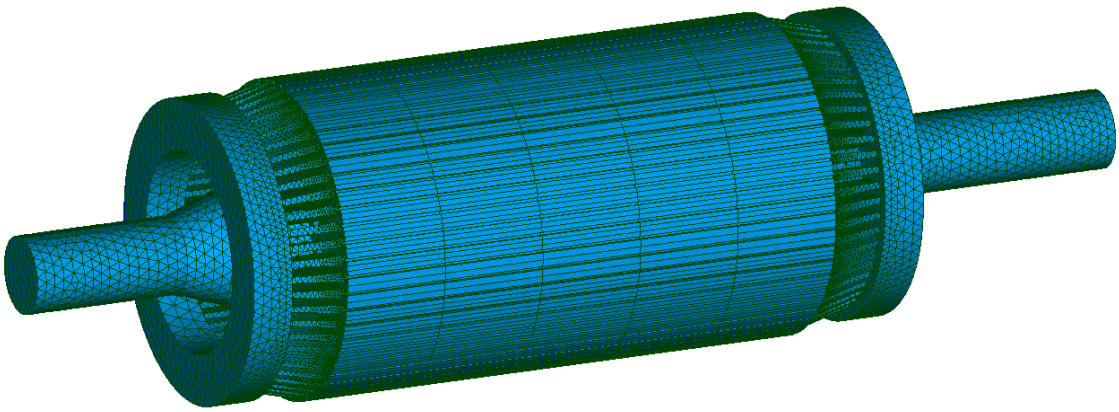

Figure 26. The test rotor with an extruded submesh along the rotor core and a tetrahedral mesh in all other regions.

The meshing of some rotor variants failed when using an extruded mesh along the rotor core. One reason for this is that *Salome* creates an edge to the inner surface of each cooling duct hole, and if the number of these holes is odd, the edges are not symmetrically aligned with respect to the vertical axis. This seemed to cause the 3D extrusion algorithm of the SMESH module to fail with the following error message: “geometry mismatches the expectation of the algorithm”. In some cases the same error message appeared even though the geometry and edges were symmetrically aligned with respect to the vertical axis. Due to this instability, the 3D extrusion meshing algorithm was not implemented in the first version of the model generator. Modification of the rotor geometries to meet the requirements of this algorithm could possibly allow implementation of the mesh extrusion

method. However, this would require further investigation and testing. Implementing the 3D extrusion method would also make the model generator more complex. It would be more advisable to use a hybrid meshing algorithm such as *MeshGems-Hybrid* which is commercial and works inside *Salome* (Distene 2016). It generates a mesh with tetrahedral and other elements instead of a full tetrahedral mesh (Distene 2016). One downside would be the licensing costs.

3 Results

In the theory section of this thesis the basics of rotordynamic analysis were reviewed especially in the field of predicting critical speeds and calculating free natural frequencies. Also the advantages and difficulties in integrating geometry, meshing and analysis were reviewed. It was found out that usually only 20% of the total time required for model creation and analysis is spent for the analysis itself (Cottrell *et al*. 2009, p. 2). This supports the idea of automating most of the routine model generation and analysis processes. The literature also supports the concept of using substructuring in rotordynamic analyses because it can speed up the solving process and increase flexibility.

As a result, an automated 3D model generator was programmed with Python to work with an open-source pre- and post-processing software, *Salome*. The model generator takes three files as input, of which only two require information to be defined by the user. In the future only one file should require user input. This software component was then integrated to work with an open-source multiphysics simulation software *Elmer* to enable solving rotor natural frequencies for three standard cases without any further user input. The whole process runs automatically through a work queue handling system which enables parallel processing with multiple processor cores.

The initial meshing parameters applied in the model generator were obtained by testing the meshing of different test case rotors. An adjustment loop was implemented to ensure successful mesh computation in case of failed attempts. One method of using a combined tetrahedral and extruded mesh instead of a full tetrahedral mesh was tested to reduce the amount of nodes and to speed up the meshing and solving process. This proved to be an unreliable method when utilizing the open-source software tools applied in this study. This alternative meshing method could possibly be implemented with more robust commercial tools.

In the first chapter 3.1 of this section geometries of two rotors, A and B, are generated with the model generator and validated by comparing them to reference geometries created with *Siemens NX*. These rotor models are then utilized in the second chapter 3.2 to compare the calculated free rotor natural frequencies to experimentally measured values. In addition, they are compared to values calculated with *Ardas*, a commercial rotordynamic analysis software which is based on a 2D model. After this, a Campbell diagram of rotor A is calculated and compared to a reference Campbell diagram calculated with *Ardas* in chapter 3.3. Adjusting the material and computation parameters to minimize the difference between the calculated and measured results was out of scope of this thesis.

3.1 Geometry validation

The geometry and mesh generation for rotor A took 2 minutes and 15 seconds and for rotor B 3 minutes and 14 seconds. The geometries and meshes were generated using the default modeling and meshing parameter values shown in Table 2, BAR_SCONTACT value 0, and the corresponding adept.dat and rotor_data.gdc files of both rotors. The parameters are explained in subchapters 2.4.1 and 2.4.2. Only one processor core was utilized in both cases, since the geometry and mesh generation were not parallelized.

Models of rotors A and B previously created with *Siemens NX* CAD software at ABB were utilized as reference models for checking the geometry validity of the models generated with *Salome*. The generated geometries are shown and compared with the reference geometries in Figure 27. Some deviations result from missing chamfers in the reference models. In the *Salome* models, all diameter changes are modeled with conical segments, which means that shaft diameter changes without chamfers are not possible. In both models, the smallest details of the real rotors were left out. These include for example small fillets or chamfers required for machining the parts.

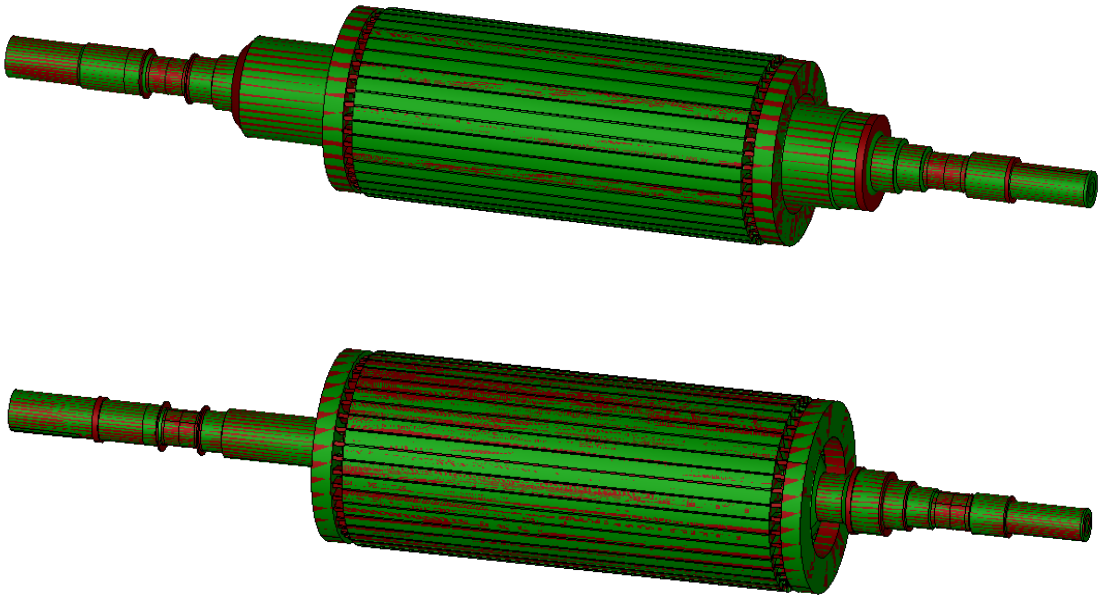

Figure 27. The reference geometries of rotor A (top) and rotor B (bottom) shown in red positioned exactly on top of the respective generated geometries shown in green. Only minor deviations between the generated and reference geometries exist.

Table 4 and Table 5 show the calculated masses of all parts of the rotor models A and B. These were obtained by using the volume data given by *Salome* and *Siemens NX*.

Percentage differences of the generated model part masses and the reference model part masses are also shown in Table 4 and Table 5.

Table 4: Comparison of the generated rotor A model mass to the reference model mass. The values with a plus or minus sign describe the percentage difference to the reference value.

Part	Reference mass [kg]	Model mass [kg]
Shaft	476.7	475.5 -0.25%
Core	467.0	468.3 +0.28%
Rings	61.4	61.4 +0.00%
Bars	125.0	128.0 +2.4%
Total	1130.1	1133.2 +0.27%

Table 5: Comparison of the generated rotor B model mass to the reference model mass. The values with a plus or minus sign describe the percentage difference to the reference value.

Part	Reference mass [kg]	Model mass [kg]
Shaft	392.6	392.5 -0.03%
Core	910.8	910.1 -0.08%
Rings	94.2	94.2 +0.00%
Bars	195.6	197.1 +0.77%
Total	1593.2	1593.9 +0.04%

Table 4 and Table 5 show that the most difference is in the bar masses. This is caused by differences in chamfering and contact modeling between the generated and reference models. However, the absolute total difference between the masses of the reference model and generated model of rotor A and B are in both cases less than 3.2 kg. The percentage difference of the total mass in both cases is less than +0.3%. Based on this validation the generated geometries may be utilized to calculate and compare the natural frequencies of both rotors to experimentally measured values. Errors in the calculated natural frequencies caused by differences in the geometry or masses can be assumed to be negligible when the effects of differences in contact modeling are omitted. The possible deviations caused by differences in contact modeling are discussed in chapter 3.2.

3.2 Comparison of free rotor natural frequencies

The mesh generation and free rotor eigenvalue computation process was executed for both rotors A and B using 32 processor cores. The total time elapsed for rotor A was 6 minutes and 20 seconds of which the time to solve was 4 minutes and 5 seconds. The total time elapsed for rotor B was 12 minutes and 58 seconds of which the time to solve was 9 minutes and 44 seconds. In both cases the number of modes solved was 15.

As a reference, modeling a rotor model manually and analyzing the free rotor eigenvalues with commercial software tools at ABB takes approximately 4 hours. The assumption is that all required information is directly available. If information such as drawings, have to be searched, the total time required is approximately 8 hours. (Kinnunen 2016).

The following simulation results shown in Table 6 and Table 7 were acquired with *Elmer* for rotor A and B respectively and compared to results from experimental measurements conducted by ABB. Since there was also simulation data of ABB from *Ardas* available for both rotors, those values were also added as references. *Ardas* is unable to calculate torsional eigenvalues since it is based on a 2D model. Therefore the reference natural frequencies of torsional modes from *Ardas* are missing. The third bending mode of rotor A is also missing since the measurement value was not available. For rotor B, measurement data of only two modes was available.

In the measurements and *Elmer* simulations the rotors are floating freely without any constraining boundary conditions and the applied material parameters are the same. In the *Ardas* simulation the rotor is floating on bearings with negligible stiffness and damping. It has a rotational speed of 20 RPM (Revolutions per minute), which is considered negligible. The mode shapes corresponding the simulated natural frequencies of *Elmer* are shown in Figures 28-31 for rotor A and in Figure 32 and Figure 33 for rotor B. In addition, the generated linear element meshes utilized for calculations can be seen from these figures.

Table 6: Comparison of the measured and calculated free natural frequencies of rotor A. The values with a plus or minus sign describe the percent difference to the corresponding measured value.

Mode	Measured [Hz]	Calculated (Elmer) [Hz]	Calculated (Ardas) [Hz]
1st bending	236.4	239.6 +1.4%	216.8 -8.3%
2nd bending	316.4	323.1 +2.1%	293.9 -7.1%
3rd bending	-	524.1	484.6
1st torsional	653.1	704.3 +7.8%	-

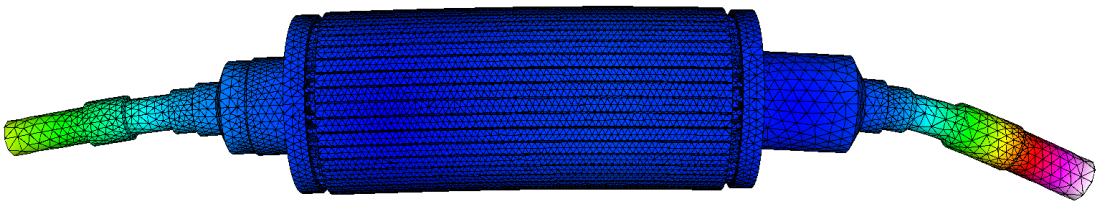

Figure 28. The 1st bending mode of rotor A.

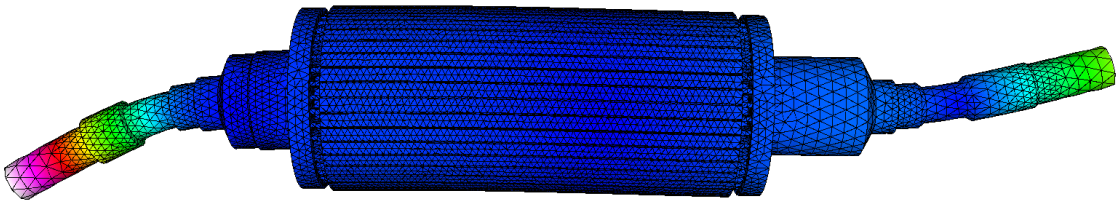

Figure 29. The 2nd bending mode of rotor A.

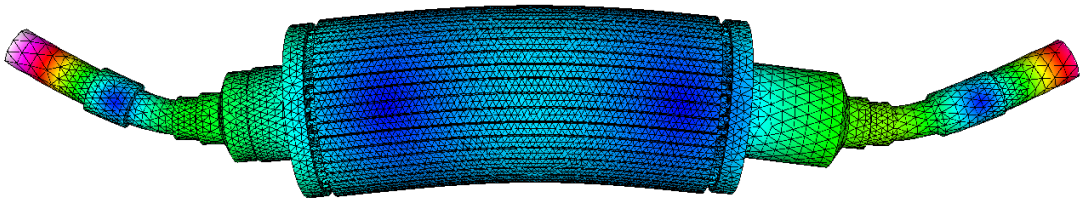

Figure 30. The 3rd bending mode of rotor A.

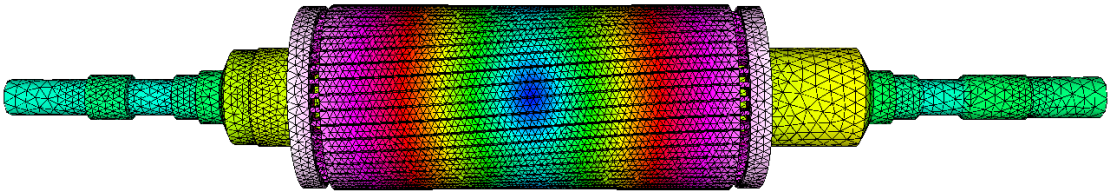

Figure 31. The 1st torsional mode of rotor A.

Table 7: Comparison of the measured and calculated free natural frequencies of rotor B. The values with a plus or minus sign describe the percent difference to the corresponding measured value.

Mode	Measured [Hz]	Calculated (Elmer) [Hz]	Calculated (Ardas) [Hz]
1st bending	129.0	140.3 +8.8%	122.4 -5.1%
2nd bending	222.0	248.3 +11.8%	208.8 -5.9%

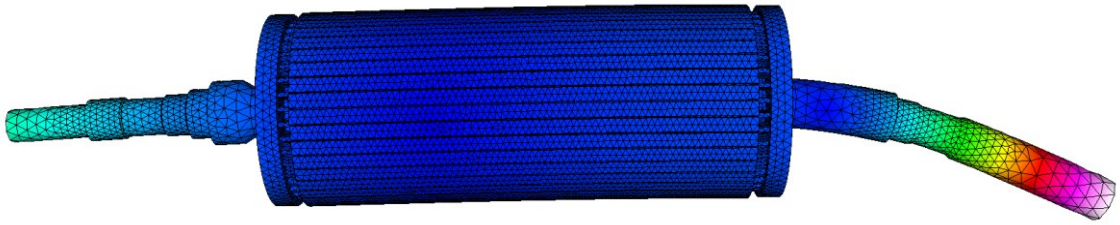

Figure 32. The 1st bending mode of rotor B.

Figure 33. The 2nd bending mode of rotor B.

The shaft of rotor A is stiffer than the shaft of rotor B due to its greater diameter near the rotor core. The shaft of rotor B is less stiff due to a smaller diameter near the rotor core. This can be seen from the figures illustrating the mode shapes. This means that the stiffness of the rotor core has a greater effect on the natural frequencies in case of rotor B compared to rotor A. When looking at the difference between the values calculated with *Elmer* and measured values in Table 6 and Table 7, it can be seen that for rotor A the differences are smaller than for rotor B. This can be partially explained by the difference in shaft stiffnesses. The natural frequencies of rotor B calculated with *Elmer* could be adjusted to match the measured values by destiffening the rotor core through material parameter adjustments.

Based on the results, the free rotor natural frequencies of the first 4 modes can be predicted with satisfactory accuracy. The results could be more accurate after adjustments to the material and simulation parameters. There are two possible sources of errors and

one aspect to be considered in these natural frequency comparisons which might cause uncertainty. The first possible reason for error in the calculated results is that the linear element mesh causes the model to be stiffer than the real rotor. This causes the natural frequencies to be greater than the measured values. Utilizing second order elements could produce more accurate results. Linear elements were utilized to keep the amount of nodes low due to memory issues when using the direct solver of *Elmer*. It requires a high amount of memory which is dependent of the amount of DOFs. Applying the iterative solver of *Elmer* might solve this problem, but setting it up demands additional work and adjustments.

The second possible source of error could be that in reality the bars do not always have contact on the upper and lower surfaces along certain lengths. These are defined by the swaging procedure in the manufacturing of rotors. (Kanninen 2006) The model generator models the bars with full top and bottom surface contact to the rotor core and only the side contact can be set off. This was decided to enable testing an extruded meshing algorithm along the rotor core, which requires the geometry to be constant in the direction of extrusion. In this case that is the direction of the z-axis which is the shaft axis. Testing of the mesh extrusion method was explained in subchapter 2.4.3. Due to top and bottom surface contact of the bars throughout the full length of the rotor core, the rotor is stiffer than in reality. Furthermore, no keyways or drillings in the shaft ends were modeled. These reasons cause the natural frequencies to be greater than in reality.

One aspect which might cause uncertainty is that the experimental measurements were conducted for a rotor supported with four rubber pads. This test arrangement should be close to a rotor floating freely, but still the supports might result in some inaccuracies. Very accurate experimental free rotor natural frequency measurement results are acquired of a rotor suspended vertically from one point (Kanninen 2006).

3.3 Comparison of Campbell diagrams

In this chapter a Campbell diagram of rotor A calculated with *Elmer* is compared to a reference diagram calculated with *Ardas*. The diagram calculated with *Ardas* was chosen as a reference, since no experimentally determined Campbell diagram was available.

This comparison was conducted to test the functioning of the computation system and to obtain data of the accuracy of the results. Furthermore, the total time required to generate a model and solve the eigenvalues of a rotating system with a full FEM model without substructuring was of interest.

The bearings in both cases were rigid and there was no viscous or structural damping present in the bearings or rotors. The rotor was pin-jointed from the bearing nodes and the material parameters were equivalent in both cases. Rigid bearings were used because defining bearing stiffnesses with the computation system to the solver input file of *Elmer* was not implemented yet. In both cases the eigenvalues were calculated with rotational speeds ranging from 0 to 6000 RPM with steps of 600 RPM to obtain data points to fit a linear curve. The reference diagram calculated with *Ardas* is shown in Figure *34*. The gyroscopic effects are small since the bearings are rigid, there is no damping, and the rotor geometry does not have large variations in the diameter. Large variations in diameters cause the rotor to be more gyroscopic as can be seen from equations (20) and (21) in subchapter 2.2.3.

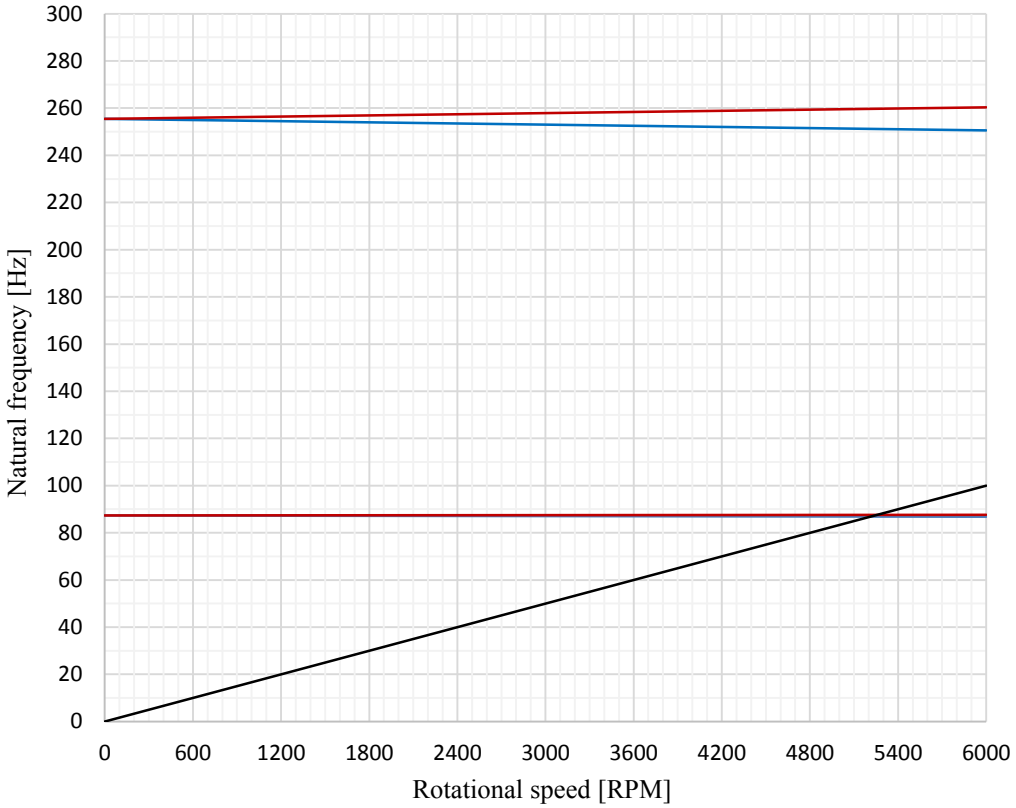

Figure 34. *The Campbell diagram of rotor A calculated with Ardas. The rotor is pin-jointed from the bearing nodes with rigid bearings and there is no damping included. The red and blue lines represent the first and second forward and backward whirling natural frequencies, respectively. The black line represents the excitation.*

The Campbell diagram shown in Figure 35 was calculated with a mesh generated with the default modeling and meshing parameters shown in Table 2 in subchapter 2.4.2. The mesh was manually modified to reduce the amount of DOFs. After this the linear elements were converted to second order elements with *Elmer*. The reduced amount of DOFs allowed applying the direct solver of *Elmer* without memory issues.

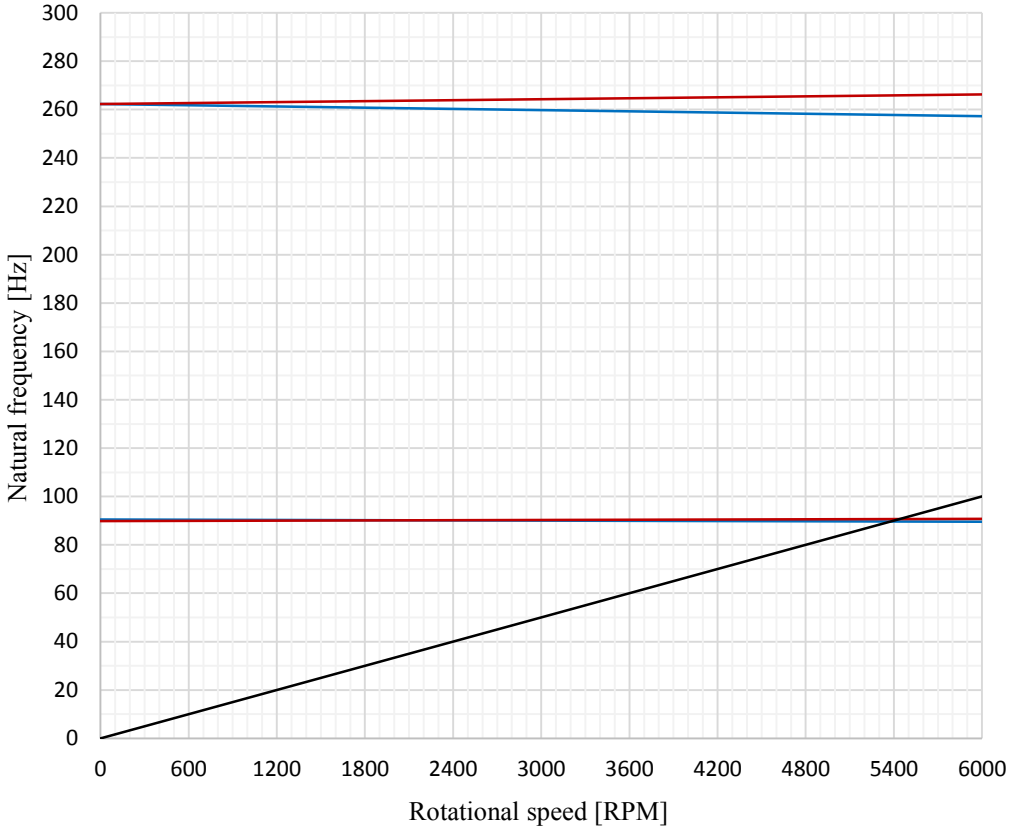

Figure 35. The Campbell diagram of rotor A calculated with Elmer. The rotor is pin-jointed from the bearing nodes with rigid bearings and there is no damping included. The red and blue lines represent the first and second forward and backward whirling natural frequencies, respectively. The black line represents the excitation.

Table 8 shows the first theoretical critical speeds defined by the intersections of the first backward whirling and excitation lines. The difference of the results is explained mainly with the difference of the models. *Ardas* is based on a 2D model while a full 3D model was utilized for *Elmer*. The modified mesh used for *Elmer* might also cause some inaccuracies.

Table 8: The first theoretical critical speed of rotor A calculated with Ardas and Elmer.

Ardas [RPM]	Elmer [RPM]	Difference [%]
5204	5383	3.4

The time to generate the model and to calculate the eigenvalues of rotor A for one rotational speed using linear elements and a mesh with 67241 nodes required approximately 30 minutes using 32 cores. Thus, solving a Campbell diagram for rotor A with rotational speed range from 0 to 6000 RPM with steps of 600 RPM requires approximately 5 hours in total. The calculation of a more accurate diagram with more steps requires even more time. This supports the idea of utilizing substructuring to speed up the computation process.

4 Summary and discussions

The primary objective was to develop a 3D model generator which automatically creates 3D finite element meshes of rotating electrical machine rotors based on product data and user-defined parameters. The generated models were utilized for eigenvalue analysis and automation was applied to the mesh generation and eigenvalue computation process of three standard cases. Usually, only 20% of the model creation and analysis time is spent for the analysis itself (Cottrell *et al*. 2009, p. 2), which supports the idea of automating and integrating geometry creation, meshing, and analysis. Automation enables the use of pace-setting technologies, such as design optimization and frees analysts' and designers' time to solve more challenging tasks. Furthermore, using 3D models enables more accurate rotordynamical simulations than those acquired with a commercial rotordynamical simulation software *Ardas*, which is based on a 2D model.

The secondary objective was to validate two generated rotor models by comparing them to reference geometries. Using these models, free rotor eigenvalues were numerically solved and compared to experimental measurement data of the corresponding rotors. In addition, the available values calculated with *Ardas* were compared to measured values as a reference. An additional objective was to calculate a Campbell diagram for one rotor model with *Elmer* and to compare it to a Campbell diagram calculated with *Ardas*.

This thesis reviewed the advantages and difficulties of integrating geometry modeling, mesh generation and analysis. In addition, the theoretical background related to rotordynamical analysis was reviewed to clarify the principles of solving eigenvalue problems of non-rotating and rotating systems with and without hysteretic damping. This thesis also discussed the calculation of stiffness matrices for isotropic and transversely isotropic materials with and without hysteretic damping, because the matrices are required for the solver input file of *Elmer*.

The principle of substructuring with component mode synthesis was presented as a method to speed up solving rotordynamical problems through the reduction of the total amount of DOFs. Based on the generated mesh and mesh topology, which contains interface DOFs, a reduced system can be generated with *Elmer*. This was not implemented as a part of this thesis, but it emphasizes the importance of consistent mesh topology management in the model generation phase since the usage of component mode synthesis is implemented later. The reduced system can then be utilized in *Modysol* to connect bearings, dampers and rotating disks for faster solving of free rotor eigenvalues and Campbell diagrams.

As a result, an automated 3D model generator was successfully created. This was achieved by utilizing the Python programming language to command *Salome*, which is a

customizable open-source software that provides a generic platform for generation of geometry and preparing it for numerical calculations. The meshing was conducted with *NETGEN*, an automatic open-source 3D tetrahedral mesh generator which is integrated inside *Salome*. The benefit of utilizing open-source softwares is that they have no licensing costs. In total, 40 possible rotor variants can be automatically generated with the created model generator if only the feasible geometry variable variations are taken into account. When the varying dimensions are considered, the amount rises significantly.

The created model generator consistently manages the mesh topology. It creates 3D element volume groups of different parts for assignment of stiffness matrices in the solver input file. It also positions nodes to standard node locations to be utilized in substructuring, post-processing and assignment of boundary conditions. This enabled transferring information from *Salome* to the solver input file of *Elmer* and automating the whole mesh generation and analysis process.

Some geometric details caused the tetrahedral mesh generation to fail unless the mesh was allowed to be unnecessarily dense. The goal was reliable mesh generation with as few nodes as possible for fast solving. Automated adjusting of meshing parameters and rule-based geometry simplifications were implemented to achieve this goal. The utilization of an extruded submesh along the rotor core was tested to further reduce the amount of nodes. However, this method was too unreliable to be implemented in the first version of the model generator.

The percent difference of two generated rotor model masses compared to reference model masses were in both cases less than +0.3%. The natural frequencies of two free rotors calculated with *Elmer* were greater than the measured values with differences ranging from +1.4% to +11.8%. Moreover, the natural frequencies of the same rotors calculated with *Ardas* were lower than the measured values with differences ranging from -5.1% to -8.3%. Adjustments to minimize the error between calculated and measured values were out of scope of this study. Even without any adjustments, the lowest natural frequencies and mode shapes can be predicted with satisfactory accuracy. The total time required for generating the model and analyzing free rotor eigenvalues was reduced to approximately one tenth when compared to full manual work.

A Campbell diagram of one rotor was calculated with a solver of *Elmer*, which takes into account the gyroscopic effects caused by rotation. This process was only automated to such extent that the eigenvalues for one rotational speed can be solved at a time. This step has to be repeated with a number of different rotational speeds to obtain enough data points to plot a Campbell diagram. The acquired diagram had only small deviations when compared to a diagram calculated with *Ardas*. The percent difference between the first

theoretical critical speed obtained with *Ardas* and *Elmer* was +3.4%. Experimental measurement data for this rotor was not available.

As a conclusion, setting up and utilizing an automated computation system with open-source software tools is a cost-efficient and flexible method to reduce manual work. Further development of the created automated computation system is recommended based on the results of this thesis.

4.1 Future work and visions

Suggestions for future work are divided into three groups:

- improvements which can be implemented directly
- improvements which require some designing and/or investments
- improvements which demand a project of a longer timespan

The improvements which can be implemented directly include creating element volume groups of the shaft portions which are inside the bearings. This enables assigning stiffer material properties to these regions to take into account the stiffening effect of bearings. Also, the effect of swedging on the contact between the rotor core and the upper and lower bar surfaces could be taken into account by modeling a gap between these parts along a certain length to remove contact. This would presumably decrease the calculated natural frequencies by destiffening the rotor. Furthermore, the post-processing of results should be automated so that they are easily available and interpretable. The whole system should be integrated to run through a web browser on the computational server. Moreover, logic to automatically choose meshing parameters based on some key dimensions or features could be implemented, especially when the system is extended to a wider amount of variations. Also, setting up and tuning an iterative solver of *Elmer* might solve the memory problems encountered when using second order elements and a higher number of DOFs. Switching from linear to second order elements presumably lowers the calculated natural frequencies and increases accuracy.

The improvements which require some designing and/or investments include creating a system to generate rotor_data.gdc files easily based on shaft drawings or data from *Adept*. Furthermore, a system to obtain material data of individual parts from adept.dat and calculate stiffness matrices based on that should be established. In the current version, there is only one set of material parameters for each part. If there is a limited amount of data available in *Adept*, a material mapping could be created so that for example, a certain group of different but similar copper alloys lead to the same material properties. Additionally, the generation of the reduced system with *Elmer* and solving Campbell diagrams with *Modysol* should be fully automated. Instead of using only one interface node at the bearing points for substructuring, all nodes on the shaft surface inside the bearing could be utilized. Furthermore, one recommendation to reduce the amount of DOFs would be utilizing a commercial hybrid meshing algorithm, such as *MeshGems-Hybrid*, which generates a mesh with tetrahedral and other elements instead of a full tetrahedral mesh.

The improvements which demand a project of a longer timespan include extending the model generator to a wider variety of rotor variants and completely different types of

rotors. In addition, extending it to other sub-assemblies of rotating electrical machines such as the stator, stator frame, bearings and foundation would require a project of a longer timespan. Since the first version of the model generator is implemented, utilizing it as a reference would accelerate that particular project.

The method of automating computational engineering could be utilized more in all well-known standard structural analyses, which are routinely repeated in companies. However, some limitations exist for this method to be utilized more widely. Four limitations are discussed next. The first limitation is that this method requires initial effort to set up the system and it might be difficult to estimate the return of investment. The second limitation is that some more complex geometries might be difficult to create and mesh with the current version of *Salome*. Geometries for analysis often require simplifications and therefore it is not a major limitation, but if the base shape of the object is complex and can not necessarily be simplified, it might become a limitation. However, *Salome* developers are constantly releasing new versions of the software with improved functionality and new features. The third limitation is that the product data has to be in a form which can be directly utilized for parametric model generation. This probably is not the case in many companies. The fourth limitation for setting up automated computation systems with open-source software is often a limited amount of user documentation and support. This means that in many cases more time is required for development, since discovering the best methods to implement the desired functionality demands more testing and research.

One further research topic could be minimizing the error between the calculated results and measured results in a wider scope by taking into account many different types of rotors. Extending the system to other analysis types such as response analysis and centrifugal pole stresses could also be a further research topic. Moreover, a further topic for reasearch could be the effect of substructuring on the accuracy and computation times of natural frequencies and Campbell diagrams.

References

ABB. 2011. High voltage induction motors. Technical catalog for IEC motors. 172 pages. [Referred 13.1.2016]. Available: https://library.e.abb.com/public/0cc54967fa9045078b6a9dd9b7fc7510/HV_Induction_9AKK103508%20EN%2006-2015.pdf.

ALSTOM. 2002. ALSTOM Rotor Dynamic Analysis System ARDAS Release 2.9 Main Calculation module User's Manual. 2002. ALSTOM Power Sweden AB. Internal document. 158 pages.

Cottrell, J. & Hughes, T. & Bazilevs, Y. 2009. Isogeometric analysis - Toward integration of CAD and FEA. John Wiley & Sons, Inc. 360 pages. ISBN: 978-0-470-74873-2.

Distene S.A.S. 2016. Volume Meshing: MeshGems-Hybrid. [Referred 9.2.2016]. Available: http://www.meshgems.com/volume-meshing-meshgems-hybrid.html.

Ewins, D. 2000. Modal Testing: Theory, Practice and Application. Second Edition. Research Studies Press. USA. 562 pages. ISBN: 0-86380-218-4.

Genta, G. 2005. Dynamics of Rotating Systems. Springer-Verlag. New York, USA. 658 pages. DOI: 10.1007/0-387-28687-X.

Inman, D. 2007. Engineering Vibration. Third Edition. Prentice-Hall, Inc. USA. 688 pages. ISBN: 978-0-132-28173-7.

Jones, M. & Price, M. & Butlin, G. 1995. Geometry Management Support for Auto-Meshing. Cambridge. England. 12 pages. [Referred 11.11.2015]. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.34.4382&rep=rep1&type=pdf.

Kanninen, P. 2006. Sähkökoneen roottoripaketin akselia jäykistävä vaikutus. Master's Thesis. Helsinki University of Technology. 67 pages.

Kinnunen, A. 2016. R&D Engineer. ABB Oy, Motors & Generators. Interview 8.2.2016.

Klinge, P. 2005. MODYSOL käyttöohje. VTT, Technical Research Centre of Finland. 37 pages.

Lee, S. 2005. A CAD-CAE integration approach using feature-based multi-resolution and multi-abstraction modelling techniques. Computer-Aided Design. Volume 37, Issue 9. p. 941-955. DOI: 10.1016/j.cad.2004.09.021.

Li, Y. & Barbič, J. 2015. Stable Orthotropic Materials. IEEE Transactions on Visualization and Computer Graphics. Vol. 21, No. 10. 6 pages. DOI: 10.1109/TVCG.2015.2448105.

Matthew, B. & Younan, A. & Allaire, P. & Cogill, R. 2010. Model Reduction Methods for Rotor Dynamic Analysis: A Survey and Review. International Journal of Rotating Machinery. Volume 2010, Article ID 273716. 17 pages. DOI: 10.1155/2010/273716.

Octave. 2015. About GNU Octave. [Referred 18.12.2015]. Available: https://www.gnu.org/software/octave/about.html.

Qu, Z. 2010. Model Order Reduction Techniques with Applications in Finite Element Analysis. Springer-Verlag London. 369 pages. ISBN: 978-1-84996-924-6.

Rodriguez, D. & Sturdza, P. 2006. A Rapid Geometry Engine for Preliminary Aircraft Design. 44th AIAA Aerospace Sciences Meeting and Exhibit. Reno, USA. 12 pages. DOI: 10.2514/6.2006-929.

Roivainen, J. 2009. Unit-Wave Response-Based Modeling of Electromechanical Noise and Vibration of Electrical Machines. Doctoral Dissertation. Helsinki University of Technology. 186 pages. ISBN: 978-951-22-9910-2.

Ryyppö, T. 2015. FCSmek in Daily Use. ABB Oy, Motors and Generators, Technology Development. Internal document. 171 pages.

Salome. 2015. SALOME 7.6.0 documentation. [Referred 14.12.2015]. Available: http://www.salome-platform.org/user-section/documentation/current-release.

Salome. 2014. SALOME 7 Brochure. [Referred 18.12.2015]. Available: http://files.salome-platform.org/Salome/Common/SALOME_7_2014_Brochure.pdf.

Shanmugam, A. & Padmanabhan, C. 2006. A fixed-free interface component mode synthesis method for rotordynamic analysis. Journal of Sound and Vibration. Volume 297, Issues 3-5. Pages 664-679. DOI: 10.1016/j.jsv.2006.04.011.

Takenaka, Y. & Nakajima, H. & Mehri, D. Application of Automatic Geometry Creation and Mesh Generation Technique for Port. 2000. FISITA World Automotive Congress. Seoul, Korea. [Referred 11.01.2016]. Available: http://210.101.116.115/fisita/pdf/A006.pdf.

Vance, J. 1987. Rotordynamics of Turbomachinery. John Wiley & Sons, Inc. USA. 388 pages. ISBN: 0-471-80258-1.

Zwinger, T. 2008. Elmer - an Open Source Finite Element Software for Multiphysical Problems. [Referred 18.12.2015]. Available: http://www.elmerfem.org/elmerwiki/images/c/c9/ElmerIntroTZ.pdf.

Appendix 1: The model generator main program

```
import sys
import salome
import os
from timeit import default_timer
from time import strftime

# Start measuring time
start_program = default_timer()

salome.salome_init()
theStudy = salome.myStudy

import GEOM
from salome.geom import geomBuilder
import math
import SMESH, SALOMEDS
from salome.smesh import smeshBuilder
from killSalomeWithPort import killMyPort

geompy = geomBuilder.New(salome.myStudy)

def Main():

      # Read product data
      print (strftime("%H:%M:%S") + ' Reading product data')
      (DI1, SLOTTYPE2, Q2, L_TOT2, DELTA, DI2, HSTOT2, BSI2, BSO2,
      BSY2, HSY2, HS2, VAC2, NROW2, N1AC2, DH1AC2, D1AC2, N2AC2,
      DH2AC2, D2AC2, N3AC2, DH3AC2, D3AC2, NRIBAC2, BRIBAC2, DIAC2,
      DOAC2, BRING2, HRING2, DRING2, LLBAR2, PRODUCT, REFERENCE) =
      Read_Product_Data('adept.dat')
      print (strftime("%H:%M:%S") + ' Product name: ' + PRODUCT)
      print (strftime("%H:%M:%S") + ' Product reference number: ' +
      REFERENCE)

      # Read user-defined parameters from the parameters file
      print (strftime("%H:%M:%S") + ' Reading user-defined
      parameters')
      (S, BAR_SCONTACT, GAP, BAR_RTOL, MaxSize, MinSize, SecondOrder,
      Optimize, UseSurfaceCurvature, FuseEdges, QuadAllowed,
      GrowthRate, SegsPerEdge, SegsPerRadius, MaxTry) =
      Read_Parameters('parameters.txt')

      # Scale dimensions to avoid numerical inaccuracies
      (DI1, L_TOT2, DELTA, DI2, HSTOT2, BSI2, BSO2, BSY2,
      HSY2, HS2, DH1AC2, D1AC2, DH2AC2, D2AC2, DH3AC2,
      D3AC2, BRIBAC2, DIAC2, DOAC2, BRING2, HRING2, DRING2,
      LLBAR2, MinSize, MaxSize) = Scale(DI1, L_TOT2, DELTA,
      DI2, HSTOT2, BSI2, BSO2, BSY2, HSY2, HS2, DH1AC2,
      D1AC2, DH2AC2, D2AC2, DH3AC2, D3AC2, BRIBAC2, DIAC2,
      DOAC2, BRING2, HRING2, DRING2, LLBAR2, MinSize, MaxSize, S)

      # Create orientation vector
      O = geompy.MakeVertex(0, 0, 0)
      OX = geompy.MakeVectorDXDYDZ(1, 0, 0)
      OY = geompy.MakeVectorDXDYDZ(0, 1, 0)
      OZ = geompy.MakeVectorDXDYDZ(0, 0, 1)
```

```
# Initialize shapes list
ShapesList = []

# Generate shaft
print (strftime("%H:%M:%S") + ' Generating shaft')
(Shaft, Rcore_Center, Standard_nodes,
Vertices) = Make_Shaft('rotor_data.gdc', S)
Shaft_props = geompy.BasicProperties(Shaft)
Shaft.SetColor(SALOMEDS.Color(0.478431, 0.478431, 0.478431))
ShapesList.append(Shaft)

# Generate rotor core
print (strftime("%H:%M:%S") + ' Generating rotor core')
(Rcore) = Make_Rcore(SLOTTYPE2, BSI2, BSO2, HS2, BSY2, HSTOT2,
DI1, DI2, DELTA, VAC2, NROW2, BRIBAC2, DOAC2, DH1AC2, DH2AC2,
DH3AC2, DIAC2, D1AC2, D2AC2, D3AC2, NRIBAC2, Q2, N1AC2, N2AC2,
N3AC2, L_TOT2)
geompy.TranslateDXDYDZ(Rcore, 0, 0, Rcore_Center+L_TOT2*0.50)
Rcore_props = geompy.BasicProperties(Rcore)
Rcore.SetColor(SALOMEDS.Color(0.223529, 0.223529, 0.223529))
ShapesList.append(Rcore)

# Generate short-circuit rings
print (strftime("%H:%M:%S") + ' Generating short-circuit rings')
(Sring) = Make_Sring(DRING2, HRING2, BRING2)
Sring_1 = geompy.TranslateDXDYDZ(Sring, 0, 0,
Rcore_Center+L_TOT2*0.50+BRING2+LLBAR2)
Sring_2 = geompy.TranslateDXDYDZ(Sring_1, 0, 0,
-L_TOT2-BRING2-2.0*LLBAR2, 1)
Sring_1_props = geompy.BasicProperties(Sring_1)
Sring_2_props = geompy.BasicProperties(Sring_2)
Sring_1.SetColor(SALOMEDS.Color(1, 0.333333, 0))
Sring_2.SetColor(SALOMEDS.Color(1, 0.333333, 0))
ShapesList.append(Sring_1)
ShapesList.append(Sring_2)

# Generate bars
print (strftime("%H:%M:%S") + ' Generating bars')
(Bar) = Make_Bar(SLOTTYPE2, BSI2, BSO2, HS2, HSY2, BSY2, HSTOT2,
DI1, DELTA, Q2, L_TOT2, LLBAR2, DRING2, HRING2, BAR_SCONTACT,
GAP, BAR_RTOL)
geompy.TranslateDXDYDZ(Bar, 0, 0.50*DI1-DELTA-HSTOT2, 0)
geompy.TranslateDXDYDZ(Bar, 0, 0,
Rcore_Center+L_TOT2*0.50+LLBAR2)
Bars = geompy.MultiRotate1DNbTimes(Bar, OZ, Q2)
Bars_props = geompy.BasicProperties(Bars)
Bars.SetColor(SALOMEDS.Color(1, 0.333333, 0))
ShapesList.append(Bars)

# Add shaft vertices to ShapesList
ShapesList.extend(Vertices)

# Remove duplicate faces between parts
print (strftime("%H:%M:%S") + ' Removing duplicate faces between
parts')
Partition_1 = geompy.MakePartition(ShapesList, [], [], [],
geompy.ShapeType["SOLID"], 0, [], 0)

# Export part volumes in SI-units
print (strftime("%H:%M:%S") + ' Exporting part volumes')
```

```
with open('part_volumes.txt', 'w') as f:
      f.write('Part volumes [cubic meters]\n')
      f.write('Shaft: ' + str(Shaft_props[2]/(S**3.0)) + '\n')
      f.write('Rotor core: ' + str(Rcore_props[2]/(S**3.0)) +
      '\n')
      f.write('Shortcircuit ring 1: ' +
      str(Sring_1_props[2]/(S**3.0)) + '\n')
      f.write('Shortcircuit ring 2: ' +
      str(Sring_2_props[2]/(S**3.0)) + '\n')
      f.write('Bars: ' + str(Bars_props[2]/(S**3.0)))

# Export rotor geometry in SI-units
print (strftime("%H:%M:%S") + ' Exporting the rotor geometry')
if os.path.exists('rotor.iges'): # delete old iges file
      os.remove('rotor.iges')
Export_Geom = geompy.MakeScaleTransform(Partition_1, O, (1/S))
geompy.ExportIGES(Export_Geom, 'rotor.iges')

# Create mesh
print (strftime("%H:%M:%S") + ' Generating mesh')
(Mesh_1, V_Shaft, V_Rcore, V_Sring_1, V_Sring_2, V_Bars,
Standard_nodes, NumberOfNodes) = Make_Mesh(Partition_1, Shaft,
Rcore, Sring_1, Sring_2, Bars, Standard_nodes, MaxSize, MinSize,
SecondOrder, Optimize, UseSurfaceCurvature, FuseEdges,
QuadAllowed, GrowthRate, SegsPerEdge, SegsPerRadius, MaxTry, S)

# Export mesh (scaled)
print (strftime("%H:%M:%S") + ' Exporting mesh with '\
+ str(NumberOfNodes) + ' nodes')
if os.path.exists('mesh.unv'): # delete old mesh file
      os.remove('mesh.unv')
Mesh_1.ExportUNV('mesh.unv', 0)

# Export standard node mapping with coordinates in SI-units
print (strftime("%H:%M:%S") + ' Exporting standard node
mapping')
if os.path.exists('standard.nodes'):
      os.remove('standard.nodes')

with open('standard.nodes', 'w') as f:
      for j in range(60):
            for i in range(5):
                  if Standard_nodes[j][0] != 'null':
                        f.write(str(Standard_nodes[j][i]) + ' ')
            if Standard_nodes[j][0] != 'null':
                  f.write('\n')

print(strftime("%H:%M:%S") + ' Finished!')

m, s = divmod(default_timer() - start_program, 60)
h, m = divmod(m, 60)
print('Total time elapsed: %d:%02d:%02d') % (h, m, s)

# Make objects visible in study (for GUI use)
geompy.addToStudy(O, 'O')
geompy.addToStudy(OX, 'OX')
geompy.addToStudy(OY, 'OY')
geompy.addToStudy(OZ, 'OZ')
geompy.addToStudy(Shaft, 'Shaft')
geompy.addToStudy(Rcore, 'Rcore')
```

```
geompy.addToStudy(Sring_1, 'Sring_1')
geompy.addToStudy(Sring_2, 'Sring_2')
geompy.addToStudy(Bars, 'Bars')
geompy.addToStudy(Partition_1, 'Partition_1')

# Kill the Salome session with the current port
killMyPort(os.getenv('NSPORT'))
```

Appendix 2: Test cases and benchmarking

Case	SLOTTYPE2	BSO2 == BSI2	BAR_SCONTACT	VAC2	NROW2	BAR_RTOL	Meshing attempts	Nodes [pcs]	Modeling [min:sec]	Free eigval [min:sec]
1	1	1	0	0	0	0.93	1	65465	1:48	3:40
2	1	1	0	2	1	0.93	1	58383	2:18	2:11
3	1	1	0	2	1	0.93	1	125087	4:04	6:26
4	1	0	0	2	1	0.93	1	126540	5:10	7:15
5	2	1	0	2	1	0.93	1	185121	7:32	12:08
6	2	1	0	2	1	0.93	3	217971	7:58	14:47
7	2	0	0	2	1	0.93	1	78529	3:23	3:50
8	2	0	0	2	1	0.93	1	101927	4:33	4:18
9	1	0	0	1	1	0.93	1	80251	3:04	3:28
10	2	0	0	1	1	0.93	2	63404	2:11	4:22
11	2	1	0	1	1	0.93	1	67239	2:15	4:05
12	2	1	0	1	1	0.93	1	108513	3:14	9:44
13	1	1	1	0	0	0.93	1	52361	1:34	2:32
14	1	1	1	2	1	0.93	1	56095	2:02	1:56
15	1	1	1	2	1	0.93	1	96666	3:31	5:22
16	1	0	1	2	1	0.93	1	96581	4:31	5:29
17	2	1	1	2	1	0.93	1	159090	7:07	14:59
18	2	1	1	2	1	0.93	1	58302	2:39	2:46
19	2	0	1	2	1	0.93	1	61279	3:06	2:38
20	2	0	1	2	1	0.93	1	88753	4:16	4:54
21	1	0	1	1	1	0.93	1	64288	2:24	3:43
22	2	0	1	1	1	0.93	2	58346	2:08	3:26
23	2	1	1	1	1	0.93	1	60720	2:09	3:11
24	2	1	1	1	1	0.93	1	96232	3:02	7:12